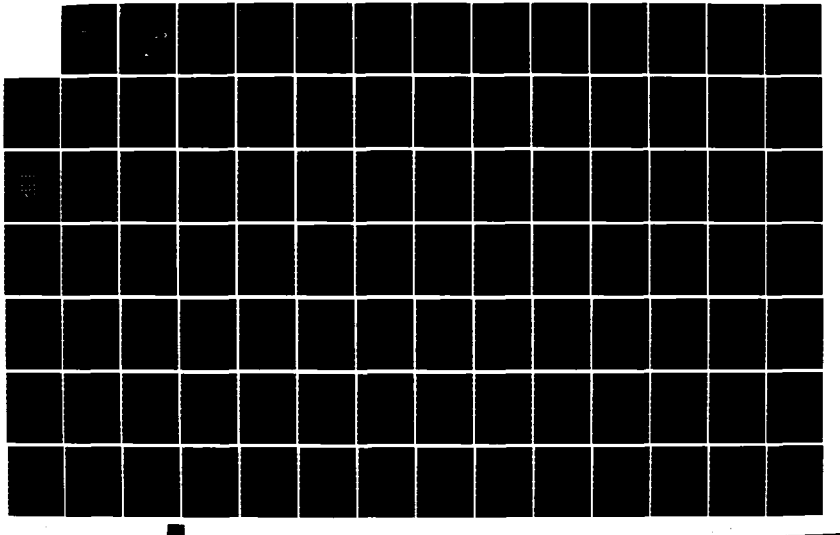
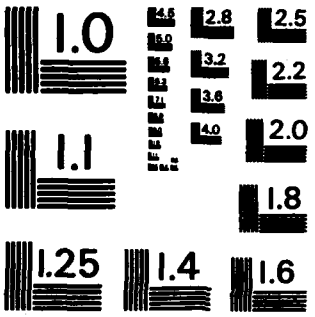
AD-A162 461 STS (SPACE TRANSPORTATION SYSTEM) TASK SIMULATOR(U) AIR FORCE ACADEMY CO S ALFANO 15 AUG 85 USAFA-TR-85-12 1/2

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

(12)

USAFA TR 85-12

STS TASK SIMULATOR

SALVATORE ALFANO, CAPTAIN, USAF

AUGUST 1985

FINAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY
COLORADO SPRINGS, CO 80840

This research report is presented as a competent treatment of the subject, worthy of publication. The United States Air Force Academy vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the author.

This report has been cleared for open publication and/or public release by the appropriate Office of Information in accordance with AFM 190-1, AFR 12-30, and AFR 80-3. There is no objection to unlimited distribution of this report to the public at large, or by DTIC to the National Technical Information Service.

This research report has been reviewed and is approved for publication.

Thomas E. McCann

THOMAS E. McCANN, Lt Col, USAF
Director of Research and
Computer Based Education

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A162461

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release. Unlimited distribution.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) USAFA TR 85-12	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Department of Astronautics 6b. OFFICE SYMBOL (If applicable) DFAS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) U.S. Air Force Academy Colorado Springs, CO 80840-5981	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) STS Task Simulator	
12. PERSONAL AUTHOR(S) SALVATORE ALFANO, Capt, USAF	
13a. TYPE OF REPORT Final report 13b. TIME COVERED FROM Aug 84 TO Aug 85	14. DATE OF REPORT (Yr., Mo., Day) 15 August 1985 15. PAGE COUNT 133
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
22	01	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Simulation, STS, Shuttle, Orbiter, Space Transportation System, Task, Rendezvous, Proximity Ops

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper describes simplified mathematical models of the Space Transportation System (STS), Reaction Control Systems (RCS), and Digital Autopilot (DAP), used in the USAFA Proximity Operations Simulator for the VAX 11/780 and the Evans and Sutherland PS 300 computers. Included in the modeling are propellant expenditures for translational maneuvers, rotational maneuvers, and attitude maintenance and on-orbit trajectory deviations induced by RCS cross coupling. This simulator serves as a learning aid for cadets studying orbital dynamics and STS mission planning and as a research platform for the Department of Astronautics.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL SALVATORE ALFANO, Capt, USAF	22b. TELEPHONE NUMBER (Include Area Code) (303) 472-4110 22c. OFFICE SYMBOL DFAS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

STS TASK SIMULATOR

by

Capt Salvatore Alfano

United States Air Force Academy
Department of Astronautics

August 15, 1985

ABSTRACT

This paper describes simplified mathematical models of the Space Transportation System (STS) Reaction Control System (RCS) and Digital Autopilot (DAP) used in the USAFA Proximity Operations Simulator for the VAX 11/780 and the Evans and Sutherland PS 300 computers. Included in the modeling are propellant expenditures for translational maneuvers, rotational maneuvers, and attitude maintenance and on-orbit trajectory deviations induced by RCS cross coupling. This simulator serves as a learning aid for cadets studying orbital dynamics and STS mission planning and as a research platform for the Department of Astronautics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The Space Transportation System (STS) Proximity Operations Simulator is a nine degrees-of-freedom trajectory integrator (six degrees of freedom for the STS and three degrees of freedom for the target) which generates digital and graphical data to describe and record relative motion of the STS Orbiter and a free-flying payload. This motion is obtained by applying the Clohessy-Wiltshire equations for terminal rendezvous/docking with the earth modeled as a uniform sphere (Appendix B) and aerodynamic forces ignored. STS position relative to target is computed by a first-order Euler integrator which uses quaternions to define the rotational state (Appendix C).

The payload is modeled as a spinning Communications satellite. The Orbiter is treated as a rigid body whose mass properties (gross weight, moments and products of inertia, and center of gravity location) are set by Program THRUSTERS and automatically read in at the beginning of the simulation. These properties remain constant for the entire simulation.

The initial state of the simulation is defined by the user. The program requires altitude and inclination of target orbit to determine proper viewing perspective and orbital dynamics. The user must also input Orbiter position relative to payload. The program sets relative velocity and rotation rates to zero and defines initial Orbiter attitude such that the payload bay faces the target with the Orbiter nose pointed away from the earth. The user also has the option of multiplying shuttle responsiveness to facilitate proximity operations training. Digital Autopilot (DAP) parameters are predefined and cannot be changed by the user.

After program initialization, user inputs are made through the hand controllers and DAP panel located in the Shuttle Aft Flight Deck Mockup. The left controller is the Translational Hand Controller (THC) and is used for positioning. The right hand controller is the Rotational Hand Controller (RHC) and controls Orbiter attitude. Cross-coupling accelerations are fully modeled in the simulation. The program reads inputs from the mockup and updates position and attitude every 40 milliseconds and monitors fuel expended from each tank. Every ten seconds Orbiter position is written to data file STS.TRK should the user wish to view his flightpath at the completion of the simulation using Program STSPATH.

2. RCS ACCELERATION MODELING

(This chapter is taken largely from chapters 3 & 4 of Reference 1)

Table 1 contains the data used by program THRUSTERS to compute forces and torques produced by individual RCS jets (with plume impingement). The left column contains array index numbers used for thruster identification. The second column contains jet identification mnemonics (not used in actual program). The next three columns contain thrust components in Orbiter body axes. These are followed by three columns containing station coordinates of thrust application point. These coordinates are used in conjunction with the last column to calculate torque about the Orbiter CG.

For a CG location defined by arbitrary station coordinates (STA_{cg}, BL_{cg}, WL_{cg}) , the torque produced by a particular jet is computed from the equation

$$\bar{L}_j = \bar{R}_j \times \bar{F}_j + C_j\bar{F}_j \tag{1}$$

where

$$\bar{F}_j = \begin{bmatrix} F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix} \tag{2}$$

$$\bar{R}_j = \begin{bmatrix} -(STA_j - STA_{cg})/12 \\ (BL_j - BL_{cg})/12 \\ -(WL_j - WL_{cg})/12 \end{bmatrix} \tag{3}$$

and where the values of F_{xj}, F_{yj}, F_{zj}, STA_j, BL_j, WL_j, and C_j are obtained from columns 3-9 of Table 1.

Table 1. RCS Thruster Data (with plume impingement)

THRUSTER NO.	THRUSTER ID	Fx (LB)	Fy (LB)	Fz (LB)	STA (IN)	BL (IN)	WL (IN)	C (FT)
1	F2F	-879.4	-26.2	119.9	306.72	14.65	392.96	0.0000
2	F3F	-879.5	0.0	122.7	306.72	0.00	394.45	0.0000
3	F1F	-879.4	26.2	119.9	306.72	-14.65	392.96	0.0000
4	F1L	-26.3	873.6	18.2	362.67	-69.50	373.73	0.0000
5	F3L	-21.0	870.3	0.5	364.71	-71.65	359.25	0.0000
6	F2R	-26.3	-873.6	18.2	362.67	69.50	373.73	0.0000
7	F4R	-21.0	-870.3	0.5	364.71	71.65	359.25	0.0000
8	F2U	-32.3	-11.7	874.4	350.93	14.39	413.46	0.0000
9	F3U	-31.9	0.0	873.5	350.92	0.00	414.53	0.0000
10	F1U	-32.3	11.7	874.4	350.93	-14.39	413.46	0.0000
11	F2D	-28.0	-616.4	-639.5	333.84	61.42	356.95	0.0000
12	F1D	-28.0	616.4	-639.5	333.84	-61.42	356.95	0.0000
13	F4D	-24.8	-612.6	-639.4	348.44	66.23	358.44	0.0000
14	F3D	-24.8	612.6	-639.4	348.44	-66.23	358.44	0.0000
15	R3A	856.8	0.0	151.1	1555.29	137.00	473.06	0.0000
16	R1A	856.8	0.0	151.1	1555.29	124.00	473.06	0.0000
17	L3A	856.8	0.0	151.1	1555.29	-137.00	473.06	0.0000
18	L1A	856.8	0.0	151.1	1555.29	-124.00	473.06	0.0000
19	L4L	0.0	870.5	-8.4	1516.06	-149.83	455.21	-0.5887
20	L2L	0.0	870.5	-8.4	1529.07	-149.83	455.21	-0.6061
21	L3L	0.0	870.5	-8.4	1542.07	-149.83	455.21	-0.6235
22	L1L	0.0	870.5	-8.4	1555.07	-149.83	455.21	-0.6410
23	R4R	0.0	-870.5	-8.4	1516.06	149.83	455.21	0.5887
24	R2R	0.0	-870.5	-8.4	1529.07	149.83	455.21	0.6061
25	R3R	0.0	-870.5	-8.4	1542.07	149.83	455.21	0.6235
26	R1R	0.0	-870.5	-8.4	1555.07	149.83	455.21	0.6410
27	L4U	29.0	72.0	870.0	1520.04	-116.51	481.65	-0.4615
28	L2U	29.0	72.0	870.0	1532.96	-116.54	481.65	-0.3725
29	L1U	29.0	72.0	870.0	1545.87	-116.58	481.65	-0.2836
30	R4U	29.0	-72.0	870.0	1520.04	116.51	481.65	0.4615
31	R2U	29.0	-72.0	870.0	1532.96	116.54	481.65	0.3725
32	R1U	29.0	-72.0	870.0	1545.87	116.58	481.65	0.2836
33	L4D	312.4	346.8	-545.7	1498.11	-101.47	420.49	1.7413
34	L2D	312.4	346.8	-545.7	1513.68	-100.61	424.63	1.4807
35	L3D	312.4	346.8	-545.7	1529.23	-99.79	428.76	1.2208
36	R4D	312.4	-346.8	-545.7	1498.11	101.47	420.49	-1.7413
37	R2D	312.4	-346.8	-545.7	1513.68	100.61	424.63	-1.4807
38	R3D	312.4	-346.8	-545.7	1529.23	99.79	428.76	-1.2208
39	F5R	-0.8	-17.0	-17.6	324.35	59.70	350.12	0.0000
40	F5L	-0.8	17.0	-17.6	324.35	-59.70	350.12	0.0000
41	R5R	0.0	-24.0	-0.6	1565.00	149.87	459.00	0.0000
42	L5L	0.0	24.0	-0.6	1565.00	-149.87	459.00	0.0000
43	R5D	0.0	0.0	-24.0	1565.00	118.00	455.44	0.0000
44	L5D	0.0	0.0	-24.0	1565.00	-118.00	455.44	0.0000

(Reproduced from Reference 1)

Table 2. Typical Orbiter Mass Properties

DESCRIPTION	VALUE
WEIGHT.....	200017.0 LB
Ixx.....	887302.0 SLUG-FT^2
Iyy.....	6386877.0 SLUG-FT^2
Izz.....	6694367.0 SLUG-FT^2
Iyz.....	-971.0 SLUG-FT^2
Izx.....	247376.0 SLUG-FT^2
Ixy.....	5622.0 SLUG-FT^2
CG STA.....	1095.3 IN
CG BL.....	0.3 IN
CG WL.....	377.4 IN

TABLE 3. RESPONSE MATRIX FOR RCS THRUSTERS

THRUSTER NUMBER	LINEAR ACCELERATION			ANGULAR ACCELERATION			PROPELLANT CONSUMPTION RATES		
	AX	AY	AZ	$\ddot{\theta}_X$	$\ddot{\theta}_Y$	$\ddot{\theta}_Z$	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK
1	-0.14157	-0.00422	0.01930	0.00009	-0.00106	-0.00010	3.10710	0.00000	0.00000
2	-0.14159	0.00000	0.01975	-0.00001	-0.00107	0.00000	3.10710	0.00000	0.00000
3	-0.14157	0.00422	0.01930	-0.00011	-0.00106	0.00009	3.10710	0.00000	0.00000
4	-0.00423	0.14064	0.00293	0.00181	-0.00017	0.00801	3.10710	0.00000	0.00000
5	-0.00338	0.14011	0.00308	0.00072	-0.00021	0.00792	3.10710	0.00000	0.00000
6	-0.00423	-0.14064	0.00293	-0.00182	-0.00018	-0.00801	3.10710	0.00000	0.00000
7	-0.00338	-0.14011	0.00308	-0.00072	-0.00021	-0.00792	3.10710	0.00000	0.00000
8	-0.00520	-0.00144	0.14077	0.00105	-0.00048	-0.00006	3.10710	0.00000	0.00000
9	-0.00514	0.00000	0.14062	-0.00199	-0.00047	0.00000	3.10710	0.00000	0.00000
10	-0.00520	0.00144	0.14077	-0.00199	-0.00048	0.00006	3.10710	0.00000	0.00000
11	-0.00451	-0.00923	-0.10295	-0.00411	0.00634	-0.00597	3.10710	0.00000	0.00000
12	-0.00451	0.00923	-0.10295	0.00435	0.00635	0.00593	3.10710	0.00000	0.00000
13	-0.00399	-0.00862	-0.10293	-0.00446	0.00622	-0.00594	3.10710	0.00000	0.00000
14	-0.00399	0.00862	-0.10293	0.00447	0.00623	0.00594	3.10710	0.00000	0.00000
15	0.13793	0.00000	0.02433	0.00155	-0.00014	-0.00140	0.00000	3.10710	0.00000
16	0.13793	0.00000	0.02433	0.00140	-0.00016	-0.00127	0.00000	3.10710	0.00000
17	0.13793	0.00000	0.02433	-0.00156	-0.00016	0.00141	0.00000	0.00000	3.10710
18	0.13793	0.00000	0.02433	-0.00141	-0.00016	0.00127	0.00000	0.00000	3.10710
19	0.00000	0.14014	-0.00135	0.00529	-0.00012	-0.00436	0.00000	0.00000	3.10710
20	0.00000	0.14014	-0.00135	0.00522	-0.00012	-0.00451	0.00000	0.00000	3.10710
21	0.00000	0.14014	-0.00135	0.00513	-0.00013	-0.00465	0.00000	0.00000	3.10710
22	0.00000	0.14014	-0.00135	0.00514	-0.00013	-0.00479	0.00000	0.00000	3.10710
23	0.00000	-0.14014	-0.00135	-0.00526	-0.00013	0.00436	0.00000	3.10710	0.00000
24	0.00000	-0.14014	-0.00135	-0.00522	-0.00014	0.00451	0.00000	3.10710	0.00000
25	0.00000	-0.14014	-0.00135	-0.00513	-0.00014	0.00465	0.00000	3.10710	0.00000
26	0.00000	-0.14014	-0.00135	-0.00514	-0.00014	0.00479	0.00000	3.10710	0.00000
27	0.00467	0.01159	0.14006	-0.00903	0.00477	-0.00073	0.00000	0.00000	3.10710
28	0.00467	0.01159	0.14006	-0.00903	0.00492	-0.00073	0.00000	0.00000	3.10710
29	0.00467	0.01159	0.14006	-0.00903	0.00506	-0.00073	0.00000	0.00000	3.10710
30	0.00467	-0.01159	0.14006	0.00904	0.00478	0.00073	0.00000	3.10710	0.00000
31	0.00467	-0.01159	0.14006	0.00904	0.00493	0.00073	0.00000	3.10710	0.00000
32	0.00467	-0.01159	0.14006	0.00904	0.00508	0.00073	0.00000	3.10710	0.00000
33	0.05029	0.05583	-0.09785	0.00687	-0.00294	-0.00123	0.00000	0.00000	3.10710
34	0.05029	0.05583	-0.09785	0.00685	-0.00308	-0.00128	0.00000	0.00000	3.10710
35	0.05029	0.05583	-0.09785	0.00694	-0.00323	-0.00133	0.00000	0.00000	3.10710
36	0.05029	-0.05583	-0.09785	-0.00698	-0.00296	0.00123	0.00000	3.10710	0.00000
37	0.05029	-0.05583	-0.09785	-0.00686	-0.00310	0.00128	0.00000	3.10710	0.00000
38	0.05029	-0.05583	-0.09785	-0.00685	-0.00324	0.00133	0.00000	3.10710	0.00000
39	-0.00013	-0.00274	-0.00283	-0.00010	0.00018	-0.00017	0.09230	0.00000	0.00000
40	-0.00013	0.00274	-0.00283	0.00010	0.00018	0.00017	0.09230	0.00000	0.00000
41	0.00000	-0.00386	-0.00010	-0.00015	0.00000	0.00013	0.00000	0.09230	0.00000
42	0.00000	0.00386	-0.00010	0.00015	0.00000	-0.00013	0.00000	0.00000	0.09230
43	0.00000	0.00000	-0.00386	-0.00027	-0.00015	-0.00001	0.00000	0.09230	0.00000
44	0.00000	0.00000	-0.00386	0.00027	-0.00015	0.00001	0.00000	0.00000	0.09230

Program THRUSTERS computes steady-state accelerations and propellant consumption rates for each of the 44 RCS thrusters and writes this information to data file RESPONSES. The user may specify a set of Orbiter mass properties or use the preprogrammed properties from Table 2.

Table 3 shows the RESPONSES data file for the properties listed in Table 2. Each row corresponds to a particular RCS thruster. The first three columns contain the body axis components of the steady-state linear acceleration vector, and columns 4-6 contain the corresponding components of angular acceleration. Columns 7-9 contain rates of propellant flow from the forward, aft left, and aft right tanks respectively.

Linear accelerations for each thruster are calculated by the equation

$$\bar{a}_j = (32.174/W) \begin{bmatrix} F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix} \tag{4}$$

where W is the Orbiter gross weight and F_{xj}, F_{yj}, and F_{zj} represent the thrust components (from Table 1). The corresponding angular accelerations are given by

$$\bar{\alpha} = [I]^{-1} \bar{L} \tag{5}$$

where

$$\bar{\alpha} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix} \tag{6}$$

$$\underline{L} = \begin{bmatrix} L_x \\ L_y \\ L_z \end{bmatrix} \tag{7}$$

$$[I] = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{zx} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{yz} & I_{zz} \end{bmatrix} \tag{8}$$

and where L_j represents the individual torque vectors produced by the designated thruster as calculated from Equations (1) through (3). The [I] matrix represents the Orbiter's moments of inertia.

Propellant flow rates (columns 7-9 of Table 3) are assumed to be 3.1071 lb/sec for each active primary jet and 0.0923 lb/sec for each active vernier jet. These rates are based on the nominal vacuum thrust magnitudes (870 lb and 24 lb) and specific impulses (290 sec and 260 sec) that are given in Reference 2. Each thruster is always assumed to be fed from its nominal source (tank) with no provisions for simulating propellant crossfeed.

3. RCS THRUSTER FIRING SELECTION

(This chapter is taken largely from chapter 4 of Reference 1)

Each iteration the program must determine which thrusters are to be fired for attitude and/or translational control. Rotation Hand Controller (RHC), Translation Hand Controller (THC), and Digital Auto Pilot (DAP) inputs are read from the shuttle mockup every 40 milliseconds and applied to one of four available firing options as commanded by the DAP. The available options are designated V (vernier jets), P (primary jets), PZI (primary jets with +Z thrusters inhibited), and HIGH Z (primary jets with additional +Z thrusters). Corresponding to each of these options is a jet-select table (Tables 4-7) which identifies the particular jet or combination of jets that is to be fired in response to each of the six possible translation acceleration commands (+X, -X, +Y, -Y, +Z, -Z) and the six possible rotational commands (+ROL, -ROL, +PCH, -PCH, +YAW, -YAW). These jet-select tables are read from the RESPONSES data file and stored in array JET. Jets are identified by the mnemonics listed in the second column of Table 1.

As indicated in Table 4 by the absence of any jet designations for the execution of translational acceleration commands, the V option can be used only for attitude control. The other three options can be used for translational and/or rotational control.

In the PZI option, no jets are fired that would expel propellant directly upward with respect to the Orbiter body. Translational acceleration in the downward direction, if commanded, is achieved (at a comparatively high propellant cost) by firing the +X and -X thrusters simultaneously. The cant angles of the +X jet and -X jet thrust lines produce a small net acceleration in the +Z (downward) direction. This option normally is used only when the Orbiter is maneuvering in the near vicinity of a payload that must be protected from jet plume impingement.

The digital auto pilot controls the attitude and translation of the Orbiter for both automatic and manned maneuvers by specifying the appropriate commands to the primary and vernier reaction control system (PRCS/VRCS) jets. The crew uses the DAP panel (Figure 1) to exercise vehicle control. The DAP panel consists of hardware pushbuttons

that allow the crew to select the translation DAP auto/manual control operations sequence; to determine whether A or B DAP configuration values will be used with auto/manual control; to select translational control; and to determine if primary or vernier jets will be commanded to fire. Appendix A contains a detailed description of the functions of the DAP control panel.

The precalculated acceleration and propellant consumption information for each thruster in the RESPONSES data file is read at the beginning of the program and stored in array JETSEL. Once the program has determined which thrusters are to be fired, the elements of the rows in JETSEL (Table 3) corresponding to those thrusters are summed. This yields net vectors for translational and rotational acceleration as well as propellant loss rate from each tank. The elements corresponding to acceleraton in array JETSEL are not recomputed to account for propellant loss during the simulation.

Table 4. Vernier (V) Jet Select Table

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+X									
-X									
+Y									
-Y									
+Z									
-Z									
+ROL	L5D								
-ROL	R5D								
+PCH	F5R	F5L							
-PCH	L5D	R5D							
+YAW	R5R								
-YAW	L5L								

Table 5. Primary (P) Jet Select Table

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+X	R1A	L1A							
-X	F2F	F1F							
+Y	F1L	L4L							
-Y	F2R	R4R							
+Z	F3U	L4U	R4U						
-Z	F1D	F2D	L4D	L2D	R4D	R2D			
+ROL	L4D	R4U							
-ROL	L4U	R4D							
+PCH	F1D	F2D	L4U	R4U					
-PCH	F3U	L4D	R4D						
+YAW	F1L	R4R							
-YAW	F2R	L4L							

Table 6. Primary with +Z Thrusters Inhibited (PZI) Jet Select Table

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+X	R1A	L1A							
-X	F2F	F1F							
+Y	F1L	L4L							
-Y	F2R	R4R							
+Z	F2F	F1F	R1A	L1A					
-Z	F1D	F2D	L4D	L2D	R4D	R2D			
+ROL	L4D								
-ROL	R4D								
+PCH	F1D	F2D							
-PCH	L4D	R4D							
+YAW	F1L	R4R							
-YAW	F2R	L4L							

Table 7. Primary with Additional +Z Thrusters (HIGH Z) Jet Select Table

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+X	R1A	L1A							
-X	F2F	F1F							
+Y	F1L	L4L							
-Y	F2R	R4R							
+Z	F2U	F3U	F1U	L4U	L2U	L1U	R4U	R2U	R1U
-Z	F1D	F2D	L4D	L2D	R4D	R2D			
+ROL	L4D	R4U							
-ROL	L4U	R4D							
+PCH	F1D	F2D	L4U	R4U					
-PCH	F3U	L4D	R4D						
+YAW	F1L	R4R							
-YAW	F2R	L4L							

4. DETERMINING POSITION AND ATTITUDE

The Orbiter translational accelerations, once computed, are added to the orbital drift accelerations. These accelerations are then integrated twice to yield velocity and position. Attitude is determined by applying rotational accelerations to the present quaternions.

The Clohessy-Wiltshire equations for terminal rendezvous/docking are used to model orbital drift. These are linearized equations of motion for an interceptor vehicle relative to a target vehicle in a circular orbit with Keplarian motion.

$$\ddot{x} = f_x' - 2\omega\dot{y} \tag{9}$$

$$\ddot{y} = f_y' + 3\omega^2 y + 2\omega\dot{x} \tag{10}$$

$$\ddot{z} = f_z' - \omega^2 z \tag{11}$$

where f_x', f_y', and f_z' are the Orbiter translational acceleration components (due to thrust), and ω is the rotation rate of the target about the planet.

The target frame is a right handed orthogonal system where x is the direction of target velocity, y is the zenith direction (along target radius vector), and z is out of orbital plane (opposite the angular momentum vector). For further explanation and derivation of Equations 9-11 refer to Appendix B.

The translational accelerations due to drift and thrust are summed in Subroutine THRUST and then integrated in Subroutine LINTEG. LINTEG is a first order Euler integrator which was chosen for fast computational speed (needed in real time simulation) in light of the fact that accelerations, and hence error, will be small.

The rotational accelerations are transformed from the body frame to the reference (target) frame by Subroutine BTOR. These accelerations are then used by Subroutine ROTATE to determine a new attitude quaternion and transformation matrix. The equations used in Subroutine ROTATE are included in Appendix C.

5. GENERATION OF VISUAL DISPLAY

The simulator uses an Evans and Sutherland PS300 computer for visual display. Three dimensional object data are loaded and stored in the mass memory of the PS300 at the beginning of the program. These data are then rotated, translated, and displayed repeatedly as commanded by the main program from the VAX 11/780.

Subroutine PS300 is responsible for loading the object data, known as vector lists, and providing a hierarchy of rotation, translation and viewing commands for later input by Subroutine LOOK. The vector lists consist of a spherical outline of the earth's continents, a star sphere, a circular horizon, target satellite, and heads-up display imagery. The reference coordinate system is the Clohessy-Wiltshire system, centered at the target and described in detail in Appendix B.

The earth's vector list is scaled at one distance unit (DU) and rotated about its axis at the rate of 15 degrees per hour. The earth is then inclined and counter-rotated a rate equivalent to the angular rate of the target orbit. Finally, it is translated in the -Y direction an amount equal to its radius plus target altitude. To complete the earth picture a horizon circle is added and the earth vector list is clipped to prevent viewing beyond the horizon.

The star vector list is triplicated and rotated 90 degrees about each axis to create a unit sphere of stars. This representation does not reflect true star positions. The sphere is scaled up by a factor of one earth radius plus twice target altitude and set counter-rotating (as was the earth). The star sphere is then translated in the -Y direction an amount equal to one DU plus target altitude and clipped so no stars appear below the horizon.

The target vector list is scaled, placed at the center of the reference coordinate system, and rotated at fifty degrees per minute to give the effect of a spinning satellite. It is not clipped and remains at the origin throughout the simulation.

Heads-up display information includes on screen printout of range, range rate, fuel used, and elapsed time in the upper left hand corner. The upper right hand corner gives an X-Y plane view of the shuttle and predicted future position in five minute increments. The increment markers

appear as small diamonds originating from the shuttle silhouette. Affixed to the target are radius, velocity, and out-of-plane arrows to facilitate quadrant determination. A fixed reticle is also displayed to help the operator judge target drift.

Subroutine LOOK uses shuttle position and attitude data relative to the target to continuously update rotation, translation, and viewing of all the predefined vector lists. Current position and attitude are used to generate three viewing vectors in the reference (target) frame: AT, FROM, and UP. AT is the line of sight vector, FROM is the position vector and UP is the overhead vector (perpendicalur to AT). The Evans and Sutherland PS300 uses these viewing vectors to scale, translate, and re-orient the stored images for perspective viewing. Figure 1 shows the relationship of the stored images to each other and how they are viewed given AT, FROM and UP vectors.

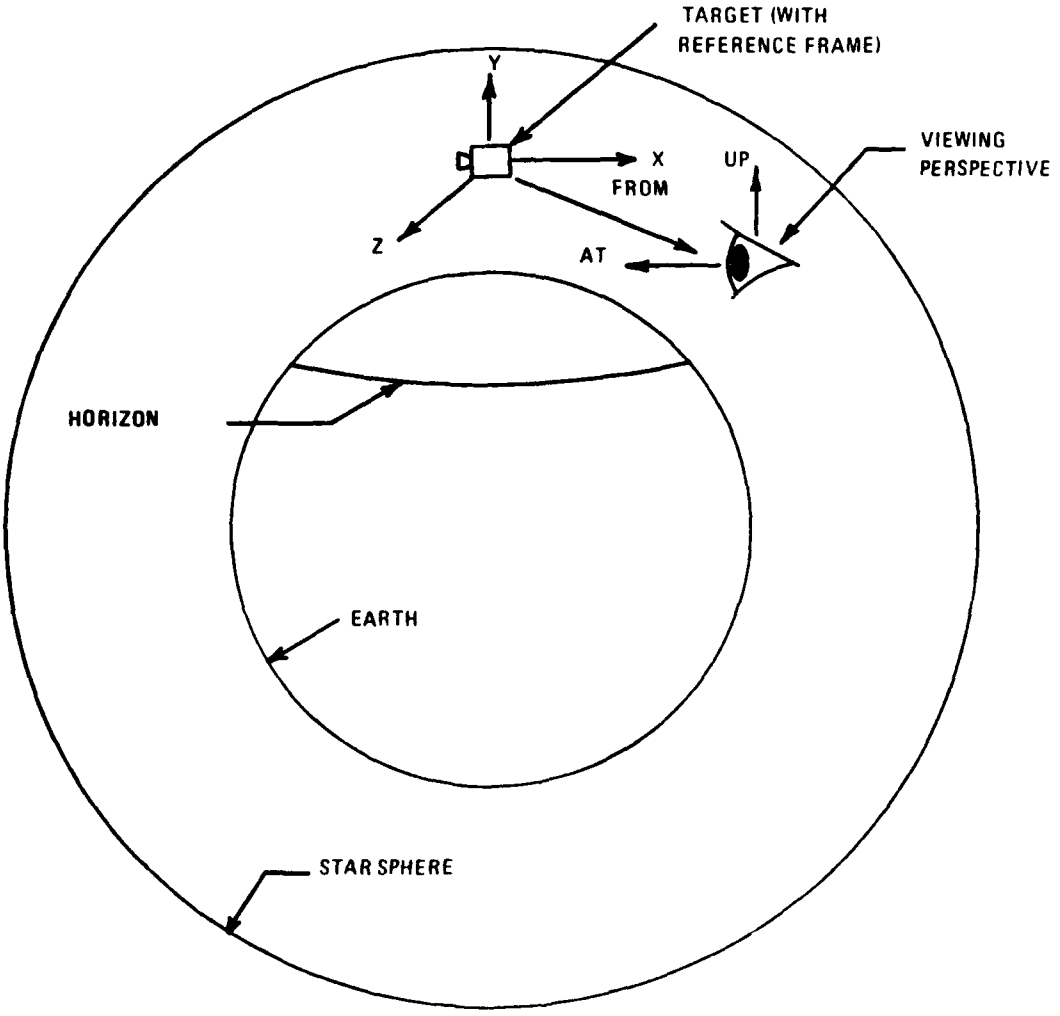


Figure 1. Visual Display Layout

REFERENCES

1. S. W. Wilson, "Engineering Description of the OMS/RCS/DAP Models used in the HP-9825A High Fidelity Relative Motion Program (HFRMP)," TRW Report No. 28415-H009-R0-00, 6 October 1978

2. Lyndon B. Johnson Space Center, "Shuttle Operational Data Book, Volume 1, Shuttle Systems Performance and Constraints Data," NASA/JSC Document No. JSC-08934 (Vol. 1), Revision A, updated 13 June 1978.

3. Lyndon B. Johnson Space Center, "Attitude and Pointing Flight Procedures Handbook", NASA/JSC Document No. JSC-10511, January 1982.

4. Astro 451 Course Handout, Astro 451, Course Readings #2, Fall 1984, Chapter F, USAF Academy, Colorado, 1984

5. Hamilton Standard Paper, Strapdown Attitude Determination Using Quaternions, Hamilton Standard Division of United Technologies, Undated

APPENDIX A

(This appendix was taken largely from Reference 3)

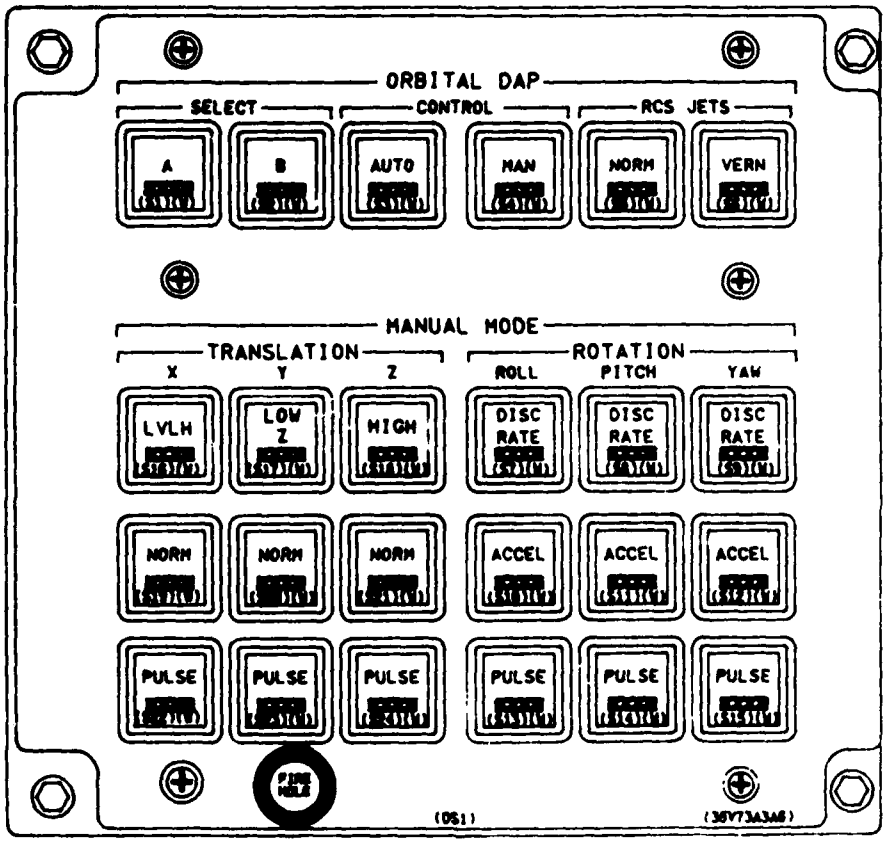


Figure 2. Panel C3: Orbital DAP Panel

AUTO/MAN Selection

AUTO A - Shuttle automatically flies directly to target.
AUTO B - Shuttle flies Clohessy-Wiltshire approach to 40 feet in front/behind and then stationkeeps.
MAN - Allows user to fly shuttle manually.

LVLH - With DISC RATE selected in all three axes and with the LVLH mode selected, the attitude that existed at the time of selection of LVLH will be held fixed within the LVLH frame until changed by use of the RHC. The RHC can be used with LVLH selected to change LVLH attitude. When the RHC returns to detent, the LVLH attitude begins to be held fixed again. This mode would be used manually for precise PROX OPS attitude control in the near vicinity of a payload.

NORM VERN JET SELECT - Primary RCS (PRCS or NORM) and vernier RCS (VRCS or VERN) jets can be used for attitude hold and maneuvers. NORM jets must be selected for translation maneuvers. With primaries selected, there is a choice of jets as shown in Tables 5-7.

Translation Submodes (NORM Jets Required)

PULSE X, Y, or Z - With this submode selected, each deflection of the THC will result in a change in velocity of the magnitude prespecified in the program. (DAP A yields .05 feet per second, DAP B yields .01 feet per second).

NORM X, Y, Z - With one of these submodes selected, continuous acceleration will occur at whatever level is available with the commanded jets for the axis and direction (+ or -) of the THC deflection.

HIGH Z - An ACCEL submode that is available for +Z translation only. Selection of this submode fires more jets than are fired for the NORM +Z command (Table 7).

LOW Z - LOW Z is a submode that takes advantage of some +Z thrust components that exist for both the +X and -X translation jets (Table 6). To get the +Z translation, the +X and -X translational jets are fired simultaneously so that the jets nearly cancel the X components of each other and combine the small Z components. The result is a small +Z velocity change. This mode was added because it provides translation away from a deployed payload without firing thrusters in the direction of the payload (thus minimizing plume impingement) as would the other +Z translation submodes. LOW Z translations also affect pitch and roll, since only downfiring jets are used. This results in significant -Z translational cross-coupling, but reduces the duty cycle for attitude maintenance.

Manual Rotational Submodes

DISC RATE - ROLL, PITCH, or YAW - When the DAP mode is manual, the submode is DISC RATE and the RHC is deflected out of the detent in an axis, jets are fired until an angular rate is achieved in that axis equal to the preprogrammed rate (DAP A yields .000873 radians per second, DAP B yields .000175 radians per second). When the rate is achieved the jets are turned off until the RHC is returned to detent in that axis unless the jets are momentarily required to maintain attitude or rates within deadbands. When the RHC is returned to detent in an axis, the attitude in that axis is snapshotted and jets are fired to stop the rate and to begin holding the snapshotted attitude.

ACCEL - ROLL, PITCH, YAW - When the DAP mode is MAN/ACCEL and the RHC is deflected out of detent, a fixed number of jets are fired to provide constant angular acceleration. When the RHC is returned to detent, the jets are turned off but the angular rates continue. The result is a free mode since the attiude is uncontrolled when the RHC is in detent 'free drift at the existing angular rates).

PULSE - ROLL, PITCH, or YAW - When the DAP mode is in MAN/PULSE and the RHC is deflected out of detent; jets are fired just long enough to achieve the preprogrammed angular rate (DAP A yields .3 radians per second squared, DAP B yields .06 radians per second squared). PULSE is also a free mode.

RHC and THC - The handcontrollers are used to input commands to the DAP, which then (depending on DAP configuration) outputs appropriate RCS jet fire commands.

APPENDIX B

(This appendix was taken wholly from Reference 4)

TERMINAL RENDEZVOUS/DOCKING

This appendix develops the equations of motion for an interceptor vehicle relative to a target vehicle in orbit about a central body when the range between vehicles is less than 8 km (5 miles).

The following approach will be taken to solve this engineering problem.

1. Establish a coordinate system:

   a. Origin at target vehicle

   b. Orthogonal right-handed system

   c. x -- In the local horizontal, in direction of target vehicle velocity vector

   d. y -- In zenith direction (along target vehicle position vector R)

   e. $\bar{R}$ -- Vector to target in fixed frame

   f. $\bar{r}$ -- Vector to interceptor in fixed frame

   g. $\bar{p} = \bar{r} - \bar{R}$ -- Position of the interceptor relative to the target.

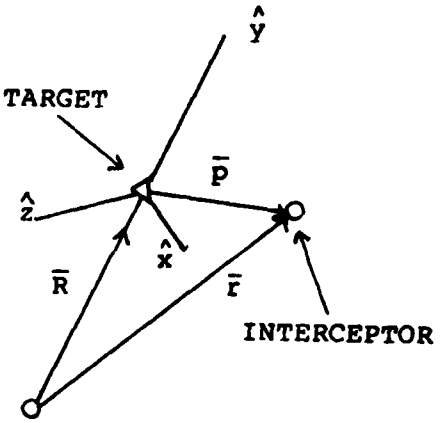


Figure 3

2. Apply $\Sigma \bar{F}_{ext} = \frac{d}{dt}(\bar{m}v)$ in the rotating coordinate system:

   a. This is a double application of the law of coriolis for derivatives in a rotating frame. (If you can't derive this, see BMW pp. 92-93.)

$$\frac{d^2(\ )}{dt^2}_F = \frac{d^2(\ )}{dt^2}_R + \dot{\bar{\omega}} \times (\ ) + \bar{\omega} \times \bar{\omega} \times (\ ) + 2\,\bar{\omega} \times \frac{d(\ )}{dt}_R$$

b. When ( ) is the position of the interceptor vehicle, $(\bar{r})$, then

$$\frac{d^2\bar{r}}{dt^2}_F = \frac{\Sigma \bar{F}_{ext}}{m} = \bar{f} = \ddot{\bar{r}}_R + \dot{\bar{\omega}} \times \bar{r} + \bar{\omega} \times \bar{\omega} \times \bar{r} + 2\,\bar{\omega} \times \dot{\bar{r}}_R$$

c. Noting the following relationships

(1) $\bar{\omega} = -\omega\,\hat{z}$

(2) $\dot{\bar{\omega}} = -\dot{\omega}\,\hat{z}$

(3) $\bar{p} = x\,\hat{x} + y\,\hat{y} + z\,\hat{z}$

(4) $\dot{\bar{p}}_R = \dot{x}\,\hat{x} + \dot{y}\,\hat{y} + \dot{z}\,\hat{z}$

(5) $\ddot{\bar{p}}_R = \ddot{x}\,\hat{x} + \ddot{y}\,\hat{y} + \ddot{z}\,\hat{z}$

(6) $\bar{R} = R\,\hat{y}$

(7) $\dot{\bar{R}}_R = \dot{R}\,\hat{y}$

(8) $\ddot{\bar{R}}_R = \ddot{R}\,\hat{y}$

(9) $\bar{r} = x\,\hat{x} + (R + y)\,\hat{y} + z\,\hat{z}$

(10) $\dot{\bar{r}}_R = \dot{x}\,\hat{x} + (\dot{R} + \dot{y})\,\hat{y} + \dot{z}\,\hat{z}$

(11) $\ddot{\bar{r}}_R = \ddot{x}\,\hat{x} + (\ddot{R} + \ddot{y})\,\hat{y} + \ddot{z}\,\hat{z}$

d. Developing the required cross products

$$\dot{\bar{\omega}} \times \bar{r} = \dot{\omega}(R + y)\,\hat{x} - \dot{\omega}x\,\hat{y}$$

$$\bar{\omega} \times \dot{\bar{r}} = \omega(\dot{R} + \dot{y})\,\hat{x} - \omega\dot{x}\,\hat{y}$$

$$\bar{\omega} \times \bar{\omega} \times \bar{r} = -\omega^2 x\,\hat{x} - \omega^2(R + y)\,\hat{y}$$

e. Now substituting from c and d into the component form of the vector equation developed in 2.b, we obtain

$$f_x = \ddot{x} + \dot{\omega}(R + y) - \omega^2 x + 2\omega(\dot{R} + \dot{y})$$

$$f_y = \ddot{R} + \ddot{y} - \dot{\omega}x - \omega^2(R + y) - 2\omega\dot{x}$$

$$f_z = \ddot{z}$$

f. Now, let $\bar{f}'$ be the specific force other than the gravitational attraction of the central body (i.e., drag, thrust, solar pressure, etc.)

$$\bar{f} = \bar{f}' + \bar{f}_g = \bar{f}' - \frac{\mu r}{r^3}$$

OR

$$f_x = f_x' - \frac{\mu x}{r^3}$$

$$f_y = f_y' - \frac{\mu(R + y)}{r^3}$$

$$f_z = f_z' - \frac{\mu z}{r^3}$$

g. Then the relative motion is described by

$$\ddot{x} = f_x' - \dot{\omega}(R + y) + \omega^2 x - 2\omega(\dot{R} + \dot{y}) - \frac{\mu x}{r^3}$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{\omega}x + \omega^2(R + y) + 2\omega\dot{x} - \frac{\mu(R + y)}{r^3}$$

$$\ddot{z} = f_z' - \frac{\mu z}{r^3} \quad \text{where} \quad r^3 = [x^2 + (R + y)^2 + z^2]^{3/2}$$

THESE ARE VERY NON-LINEAR, BUT EXACT EQUATIONS OF MOTION.

3. Linearize the equations to find an analytic solution:

$$r^3 = R^3 [(\frac{x}{R})^2 + (1 + \frac{y}{R})^2 + (\frac{z}{R})^2]^{3/2}$$

$$= R^3 [1 + \frac{2y}{R} + (\frac{x}{R})^2 + (\frac{y}{R})^2 + (\frac{z}{R})^2]^{3/2}$$

a. Since x, y and z are small compared with R,

$$r^3 \simeq R^3(1 + \frac{2y}{R})^{3/2}$$

b. Then the last terms in the equations of 2.g. take on the form

$$-\frac{\mu}{r^3} \simeq -\frac{\mu}{R^3\left(1+\frac{2y}{R}\right)^{3/2}}$$

c. Since the term $(\frac{y}{R})$ is small (less than 0.00125), use the binominal expansion to begin to simplify these equations

$$(1+\epsilon)^{-3/2} = 1 - \frac{3}{2}\epsilon + \frac{15}{8}\epsilon^2 - \ldots$$

d. Then, substituting into 2g and neglecting the small high order terms,

$$-\frac{\mu x}{R^3\left(1+\frac{2y}{R}\right)^{-3/2}} \simeq -\frac{\mu x}{R^3} + \frac{3\mu xy}{R^4} \quad , \text{ etc.}$$

So that

$$\ddot{x} = f_x' - \dot{\omega}(R+y) + \omega^2 x - 2\omega(\dot{R}+\dot{y}) - \frac{\mu x}{R^3} + \frac{3\mu xy}{R^4}$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{\omega}x + \omega^2(R+y) + 2\omega\dot{x} - \frac{\mu}{R^2} + \frac{2\mu y}{R^3} + \frac{3\mu y^2}{R^4}$$

$$\ddot{z} = f_z' - \frac{\mu z}{R^3} + \frac{3\mu zy}{R^4}$$

e. The only nonlinear terms are the last ones in the equations. These terms are smaller than the proceeding terms by $(\frac{y}{R})$. Neglecting these terms results in

$$\ddot{x} = f_x' - \dot{\omega}(R+y) + (\omega^2 - \frac{\mu}{R^3})x - 2\omega(\dot{R}+\dot{y})$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{\omega}x + (\omega^2 - \frac{\mu}{R^3})R + (\omega^2 + \frac{2\mu}{R^3})y + 2\omega\dot{x}$$

$$\ddot{z} = f_z' - \frac{\mu}{R^3} z$$

4. Now ASSUME the target vehicle is in a CIRCULAR orbit.

   a. Then

$$\omega = \frac{v_{CS}}{R} = \sqrt{\frac{\mu}{R^3}} \quad , \quad \omega^2 = \frac{\mu}{R^3} \quad , \quad \dot{\omega} = 0 \quad , \quad \dot{R} \quad \text{and} \quad \ddot{R} = 0$$

   b. The equations of motion become

$$\ddot{x} = f_x' - 2\omega\dot{y}$$

$$\ddot{y} = f_y' + 3\omega^2 y + 2\omega\dot{x}$$

$$\ddot{z} = f_z' - \omega^2 z$$

THESE ARE THE LINEARIZED EQUATIONS OF MOTION FOR AN INTERCEPTOR VEHICLE RELATIVE TO A TARGET VEHICLE IN A CIRCULAR ORBIT WITH KEPLARIAN MOTION.

APPENDIX C

(This appendix was taken wholly from Reference 5)

ATTITUDE DETERMINATION USING QUATERNIONS

The quaternion q describing the orientation of a body with respect to a reference coordinate frame may be found by integrating the quaternion differential equation:

$$\dot{q} = q \frac{\omega}{2} \tag{1}$$

where

$$q = \hat{i}q_1 + \hat{j}q_2 + \hat{k}q_3 + q_4$$

$$\omega = \hat{i}\omega_X + \hat{j}\omega_Y + \hat{k}\omega_Z$$

ω = rate of rotation of body with respect to the reference frame (in body coordinate frame).

Expansion of (1) yields the following form:

$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & \omega_Z & -\omega_Y & \omega_X \\ -\omega_Z & 0 & \omega_X & \omega_Y \\ \omega_Y & -\omega_X & 0 & \omega_Z \\ -\omega_X & -\omega_Y & -\omega_Z & 0 \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{2}$$

The quaternion components in the body (X, Y, Z) coordinate frame are

$$q_1 = \frac{u_X}{u} \sin \frac{ut}{2}$$

$$q_2 = \frac{u_Y}{u} \sin \frac{ut}{2} \qquad (3)$$

$$q_3 = \frac{u_Z}{u} \sin \frac{ut}{2}$$

$$q_4 = \cos \frac{ut}{2}$$

where

$$u = \sqrt{u_X^2 + u_Y^2 + u_Z^2}$$

A vector is transformed from body to reference coordinates by

$$\bar{x}^R = q\bar{x}^B q^* \qquad (4)$$

where

$$q^* = -\hat{i}q_1 - \hat{j}q_2 - \hat{k}q_3 + q_4$$

$$= \text{conjugate of } q$$

$$qq^* = q^* q = 1$$

The equivalent equation in vector-matrix notation is

$$\bar{x}^R = \left[T_B^R\right] \bar{x}^B \qquad (5)$$

If (4) and (5) are expanded and compared, it is seen that

$$\left[T_B^R\right] = \begin{bmatrix} q_1^2-q_2^2-q_3^2+q_4^2 & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_1q_2+q_3q_4) & -q_1^2+q_2^2-q_3^2+q_4^2 & 2(q_2q_3-q_1q_4) \\ 2(q_1q_3-q_2q_4) & 2(q_2q_3+q_1q_4) & -q_1^2-q_2^2+q_3^2+q_4^2 \end{bmatrix} \quad (6)$$

From (6), the quaternion components, expressed as functions of the matrix elements, are

$$\begin{aligned} q_1 &= \frac{1}{4q_4}(T_{32} - T_{23}) \\ q_2 &= \frac{1}{4q_4}(T_{13} - T_{31}) \\ q_3 &= \frac{1}{4q_4}(T_{21} - T_{12}) \\ q_4 &= \frac{1}{2}\sqrt{1 + T_{11} + T_{22} + T_{33}} \end{aligned} \quad (7)$$

APPENDIX D

THRUSTERS PROGRAM LISTING

```
C       THIS PROGRAM GENERATES ROTATIONAL AND TRANSLATIONAL ACCELERATIONS
C       AND ASSOCIATED FUEL COSTS FOR EACH THRUSTER AND STORES THEM IN
C       ARRAY "THRUST".  TO DO THIS IT APPLIES EACH THRUSTER'S FORCE
C       AND POSITION ON SHUTTLE BODY (STORED IN ARRAY 'JET') TO SHUTT
C       MASS, CG, AND MOMENTS OF INERTIA.
C       FOR A DETAILED EXPLANATION OF EQUATIONS/PARAMETERS REFER TO
C       TRW PAPER "ENGINEERING DESCRIPTION OF THE OMS/RCS/DAP MODELS
C       USED IN THE HP-9825A HIGH FIDELITY RELATIVE MOTION PROGRAM (HFRMP)"
C       6 OCT 78.     OR SEE CAPT. ALFANO, DFAS
C
        REAL MASS,STACG,BLCG,WLCG,RX,RY,RZ
        REAL MOI(3,3),IMOI(3,3),JET(44,12),THRUST(44,9)
        REAL VUN(12),PUN(12),PZIUN(12),PZHI(12)
        INTEGER I,J,TYPE,TABLE(4,12,9),IOSTAT
C
C
C         SELECTION TABLES ARE AS FOLLOWS:
C
        DATA (TABLE(1,1,I),I=1,9)/9*0/
        DATA (TABLE(1,2,I),I=1,9)/9*0/
        DATA (TABLE(1,3,I),I=1,9)/9*0/
        DATA (TABLE(1,4,I),I=1,9)/9*0/
        DATA (TABLE(1,5,I),I=1,9)/9*0/
        DATA (TABLE(1,6,I),I=1,9)/9*0/
        DATA (TABLE(1,7,I),I=1,9)/44,8*0/
        DATA (TABLE(1,8,I),I=1,9)/43,8*0/
        DATA (TABLE(1,9,I),I=1,9)/39,40,7*0/
        DATA (TABLE(1,10,I),I=1,9)/44,43,7*0/
        DATA (TABLE(1,11,I),I=1,9)/41,8*0/
        DATA (TABLE(1,12,I),I=1,9)/42,8*0/
C
        DATA (TABLE(2,1,I),I=1,9)/16,18,7*0/
        DATA (TABLE(2,2,I),I=1,9)/1,3,7*0/
        DATA (TABLE(2,3,I),I=1,9)/4,19,7*0/
        DATA (TABLE(2,4,I),I=1,9)/6,23,7*0/
        DATA (TABLE(2,5,I),I=1,9)/9,27,30,6*0/
        DATA (TABLE(2,6,I),I=1,9)/12,11,33,34,36,37,3*0/
        DATA (TABLE(2,7,I),I=1,9)/33,30,7*0/
        DATA (TABLE(2,8,I),I=1,9)/27,36,7*0/
        DATA (TABLE(2,9,I),I=1,9)/12,11,27,30,5*0/
        DATA (TABLE(2,10,I),I=1,9)/9,33,36,6*0/
        DATA (TABLE(2,11,I),I=1,9)/4,23,7*0/
        DATA (TABLE(2,12,I),I=1,9)/6,19,7*0/
C
        DATA (TABLE(3,1,I),I=1,9)/16,18,7*0/
        DATA (TABLE(3,2,I),I=1,9)/1,3,7*0/
        DATA (TABLE(3,3,I),I=1,9)/4,19,7*0/
        DATA (TABLE(3,4,I),I=1,9)/6,23,7*0/
        DATA (TABLE(3,5,I),I=1,9)/1,3,16,18,5*0/
        DATA (TABLE(3,6,I),I=1,9)/12,11,33,34,36,37,3*0/
        DATA (TABLE(3,7,I),I=1,9)/33,8*0/
        DATA (TABLE(3,8,I),I=1,9)/36,8*0/
        DATA (TABLE(3,9,I),I=1,9)/12,11,7*0/
        DATA (TABLE(3,10,I),I=1,9)/33,36,7*0/
        DATA (TABLE(3,11,I),I=1,9)/4,23,7*0/
        DATA (TABLE(3,12,I),I=1,9)/6,19,7*0/
C
        DATA (TABLE(4,1,I),I=1,9)/16,18,7*0/
        DATA (TABLE(4,2,I),I=1,9)/1,3,7*0/
        DATA (TABLE(4,3,I),I=1,9)/4,19,7*0/
```

```
      DATA (TABLE(4,4,I),I=1,9)/6,23,7*0/
      DATA (TABLE(4,5,I),I=1,9)/8,9,10,27,28,29,30,31,32/
      DATA (TABLE(4,6,I),I=1,9)/12,11,33,34,36,37,3*0/
      DATA (TABLE(4,7,I),I=1,9)/33,30,7*0/
      DATA (TABLE(4,8,I),I=1,9)/27,36,7*0/
      DATA (TABLE(4,9,I),I=1,9)/12,11,27,30,5*0/
      DATA (TABLE(4,10,I),I=1,9)/9,33,36,6*0/
      DATA (TABLE(4,11,I),I=1,9)/4,23,7*0/
      DATA (TABLE(4,12,I),I=1,9)/6,19,7*0/
C
C       ASSIGN VALUES TO JET ARRAY
C
      DATA (JET(1,I),I=1,9)/-879.4,-26.2,119.9,306.72,14.65,392.96,
     *0,3.1071,1/
      DATA (JET(2,I),I=1,9)/-879.5,0.0,122.7,306.72,0.0,394.45,0.0,
     *3.1071,1/
      DATA (JET(3,I),I=1,9)/-879.4,26.2,119.9,306.72,-14.65,392.96,
     *0.0,3.1071,1/
      DATA (JET(4,I),I=1,9)/-26.3,873.6,18.2,362.67,-69.5,373.73,
     *0.0,3.1071,1/
      DATA (JET(5,I),I=1,9)/-21.0,870.3,.5,364.71,-71.65,359.25,0.0,
     *3.1071,1/
      DATA (JET(6,I),I=1,9)/-26.3,-873.6,18.2,362.67,69.5,373.73,
     *0.0,3.1071,1/
      DATA (JET(7,I),I=1,9)/-21.0,-870.3,0.5,364.71,71.65,359.25,
     *0.0,3.1071,1/
      DATA (JET(8,I),I=1,9)/-32.3,-11.7,874.4,350.93,14.39,413.46,
     *0.0,3.1071,1/
      DATA (JET(9,I),I=1,9)/-31.9,0.0,873.5,350.92,0.0,414.53,
     *0.0,3.1071,1/
      DATA (JET(10,I),I=1,9)/-32.3,11.7,874.4,350.93,-14.39,413.46,
     *0.0,3.1071,1/
      DATA (JET(11,I),I=1,9)/-28.0,-616.4,-639.5,333.84,61.42,356.95,
     *0.0,3.1071,1/
      DATA (JET(12,I),I=1,9)/-28.0,616.4,-639.5,333.84,-61.42,356.95,
     *0.0,3.1071,1/
      DATA (JET(13,I),I=1,9)/-24.8,-612.6,-639.4,348.44,66.23,358.44,
     *0.0,3.1071,1/
      DATA (JET(14,I),I=1,9)/-24.8,612.6,-639.4,348.44,-66.23,358.44,
     *0.0,3.1071,1/
      DATA (JET(15,I),I=1,9)/856.8,0.0,151.1,1555.29,137.0,473.06,
     *0.0,3.1071,2/
      DATA (JET(16,I),I=1,9)/856.8,0.0,151.1,1555.29,124.0,473.06,
     *0.0,3.1071,2/
      DATA (JET(17,I),I=1,9)/856.8,0.0,151.1,1555.29,-137.0,473.06,
     *0.0,3.1071,3/
      DATA (JET(18,I),I=1,9)/856.8,0.0,151.1,1555.29,-124.0,473.06,
     *0.0,3.1071,3/
      DATA (JET(19,I),I=1,9)/0.0,870.5,-8.4,1516.06,-149.83,455.21,
     *-0.5887,3.1071,3/
      DATA (JET(20,I),I=1,9)/0.0,870.5,-8.4,1529.07,-149.83,455.21,
     *-0.6061,3.1071,3/
      DATA (JET(21,I),I=1,9)/0.0,870.5,-8.4,1542.07,-149.83,455.21,
     *-0.6235,3.1071,3/
      DATA (JET(22,I),I=1,9)/0.0,870.5,-8.4,1555.07,-149.83,455.21,
     *-0.6410,3.1071,3/
      DATA (JET(23,I),I=1,9)/0.0,-870.5,-8.4,1516.06,149.83,455.21,
     *0.5887,3.1071,2/
      DATA (JET(24,I),I=1,9)/0.0,-870.5,-8.4,1529.07,149.83,455.21,
     *0.6061,3.1071,2/
```

```
      DATA (JET(25,I),I=1,9)/0.0,-870.5,-8.4,1542.07,149.83,455.21,
     *0.6235,3.1071,2/
      DATA (JET(26,I),I=1,9)/0.0,-870.5,-8.4,1555.07,149.83,455.21,
     *0.641,3.1071,2/
      DATA (JET(27,I),I=1,9)/29.0,72.0,870.0,1520.04,-116.51,481.65,
     *-0.4615,3.1071,3/
      DATA (JET(28,I),I=1,9)/29.0,72.0,870.0,1532.96,-116.54,481.65,
     *-0.3725,3.1071,3/
      DATA (JET(29,I),I=1,9)/29.0,72.0,870.0,1545.87,-116.58,481.65,
     *-0.2836,3.1071,3/
      DATA (JET(30,I),I=1,9)/29.0,-72.0,870.0,1520.04,116.51,481.65,
     *0.4615,3.1071,2/
      DATA (JET(31,I),I=1,9)/29.0,-72.0,870.0,1532.96,116.54,481.65,
     *0.3725,3.1071,2/
      DATA (JET(32,I),I=1,9)/29.0,-72.0,870.0,1545.87,116.58,481.65,
     *0.2836,3.1071,2/
      DATA (JET(33,I),I=1,9)/312.4,346.8,-545.7,1498.11,-101.47,420.49,
     *1.7413,3.1071,3/
      DATA (JET(34,I),I=1,9)/312.4,346.8,-545.7,1513.68,-100.61,424.63,
     *1.4807,3.1071,3/
      DATA (JET(35,I),I=1,9)/312.4,346.8,-545.7,1529.23,-99.79,428.76,
     *1.2208,3.1071,3/
      DATA (JET(36,I),I=1,9)/312.4,-346.8,-545.7,1498.11,101.47,420.49,
     *-1.7413,3.1071,2/
      DATA (JET(37,I),I=1,9)/312.4,-346.8,-545.7,1513.68,100.61,424.63,
     *-1.4807,3.1071,2/
      DATA (JET(38,I),I=1,9)/312.4,-346.8,-545.7,1529.23,99.79,428.76,
     *-1.2208,3.1071,2/
      DATA (JET(39,I),I=1,9)/-0.8,-17.0,-17.6,324.35,59.7,350.12,
     *0.0,.0923,1/
      DATA (JET(40,I),I=1,9)/-0.8,17.0,-17.6,324.35,-59.7,350.12,
     *0.0,.0923,1/
      DATA (JET(41,I),I=1,9)/0.0,-24.0,-0.6,1565.0,149.87,459.0,
     *0.0,.0923,2/
      DATA (JET(42,I),I=1,9)/0.0,24.0,-0.6,1565.0,-149.87,459.0,
     *0.0,.0923,3/
      DATA (JET(43,I),I=1,9)/0.0,0.0,-24.0,1565.0,118.0,455.44,
     *0.0,.0923,2/
      DATA (JET(44,I),I=1,9)/0.0,0.0,-24.0,1565.0,-118.0,455.44,
     *0.0,.0923,3/
C
C       REASSIGN VALUES TO JET
C
      DO 900 I=1,44
        JET(I,11)=JET(I,8)
        JET(I,12)=JET(I,9)
900   CONTINUE
C
      MASS=200017.0/32.2
      MOI(1,1)=887302.00
      MOI(2,2)=6386877.0
      MOI(3,3)=6694367.0
      MOI(1,2)=-5622.0
      MOI(2,1)=-5622.0
      MOI(2,3)=971.0
      MOI(3,2)=971.0
      MOI(1,3)=-247376.0
      MOI(3,1)=-247376.0
      STACG=1095.3
      BLCG=0.3
```

```
      WLCG=377.4
C
      PRINT *,' DO YOU WISH TO USE NOMINAL SHUTTLE PARAMETERS?'
      PRINT *,' TYPE  1 FOR YES, 2 FOR NO'
      READ *,TYPE
      IF (TYPE .EQ. 2) THEN
        PRINT *,' ENTER SHUTTLE MASS (LBS)'
        READ*,MASS
C         CONVERT TO SLUGS
          MASS=MASS/32.2
        PRINT *,' ENTER MOMENTS OF INERTIA (SLUG-FT**2)'
        PRINT *,' IXX,IYY,IZZ (BODY FRAME)'
        READ*,MOI(1,1),MOI(2,2),MOI(3,3)
        PRINT *,' ENTER IYZ,IZX,IXY'
        READ*,MOI(2,3),MOI(3,1),MOI(1,2)
          MOI(2,3)=-MOI(2,3)
          MOI(3,1)=-MOI(3,1)
          MOI(1,2)=-MOI(1,2)
          MOI(3,2)=MOI(2,3)
          MOI(1,3)=MOI(3,1)
          MOI(2,1)=MOI(1,2)
        PRINT *,' ENTER SHUTTLE CG (INCHES)'
        PRINT *,' CG STA, CG BL, CG WL'
        READ *,STACG,BLCG,WLCG
      ENDIF
      PRINT *,' ******* ECOCHECK OF SHUTTLE DATA *******'
      PRINT *,' SHUTTLE MASS = ',MASS*32.2
      PRINT *,' IXX = ',MOI(1,1)
      PRINT *,' IYY = ',MOI(2,2)
      PRINT *,' IZZ = ',MOI(3,3)
      PRINT *,' IYZ = ',-MOI(2,3)
      PRINT *,' IZX = ',-MOI(3,1)
      PRINT *,' IXY = ',-MOI(1,2)
      PRINT *,' CG STA = ',STACG
      PRINT *,' CG BL = ',BLCG
      PRINT *,' CG WL = ',WLCG
C
C     GIVEN THE SHUTTLE CG LOCATIONS (STACG,BLCG,WLCG IN THE SHUTTLE
C     FRAME) THIS ROUTINE COMPUTES TORQUES IN BODY AXIS (LX,LY,LZ)
C     AND STORES THEM IN COLUMNS 8,9,10 OF ARRAY JET.  EACH THRUSTER
C     HAS ONE ROW OF ARRAY JET ASSIGNED.  COLUMNS ARE AS FOLLOWS:
C      1-FX     2-FY     3-FZ     4-STA     5-BL      6-WL
C      7-C      8-LX     9-LY     10-LZ     11-PROPELLANT FLOW RATE
C      12-TANK(1=FORWARD,2=RIGHT,3=LEFT)
C
C
      DO 175 I=1,44
        RX=-(JET(I,4)-STACG)/12.0
        RY=+(JET(I,5)-BLCG)/12.0
        RZ=-(JET(I,6)-WLCG)/12.0
        JET(I,8)=RY*JET(I,3)-RZ*JET(I,2)+JET(I,1)*JET(I,7)
        JET(I,9)=RZ*JET(I,1)-RX*JET(I,3)+JET(I,2)*JET(I,7)
        JET(I,10)=RX*JET(I,2)-RY*JET(I,1)+JET(I,3)*JET(I,7)
175   CONTINUE
C
C     ESTABLISH UNCOMPENSATED ACCELERATION PROFILES
C      MOI=MOMENT OF INERTIA MATRIX
C      IMOI=INVERSE OF MOI
C      VUN( )=VERNIER/UNCOMPENSATED RESPONSES
C      PUN( )=PRIMARY/UNCOMPENSATED RESPONSES
```

```
C       PZIUN( )=PRIMARY WITH +Z INHIBITED/UNCOMPENSATED RESPONSES
C       PZHI( )=PRIMARY WITH HIGH Z/UNCOMPENSATED RESPONSES
C
        CALL INVERT(MOI,IMOI)
        CALL JETSEL(IMOI,JET,MASS,THRUST,VUN,PUN,PZIUN,PZHI,TABLE)
C
C       OPEN FILES HERE
C
        OPEN(UNIT=2,NAME='[ALFANO.DATA]RESPONSES.DAT',TYPE='NEW',
       *FORM='UNFORMATTED',INITIALSIZE=40)
C       OPEN(UNIT=3,FILE='RSP.TBL',STATUS='NEW',
C      * IOSTAT=ISTAT)
C
C       WRITE TO THE FILE
C
105     FORMAT(7X,I2,9(3X,F9.5))
        DO 825 I=1,44
          WRITE(2)(THRUST(I,J),J=1,9)
C          WRITE(3,105)I,(THRUST(I,J),J=1,9)
825     CONTINUE
        DO 850 I=1,12
          WRITE(2)(TABLE(1,I,J),J=1,9)
          WRITE(2)(TABLE(2,I,J),J=1,9)
          WRITE(2)(TABLE(3,I,J),J=1,9)
          WRITE(2)(TABLE(4,I,J),J=1,9)
          WRITE(2)VUN(I)
          WRITE(2)PUN(I)
          WRITE(2)PZIUN(I)
          WRITE(2)PZHI(I)
850     CONTINUE
C
C       CLOSE FILES
C
        CLOSE(UNIT=2)
C       CLOSE(UNIT=3)
C
        END
C
C
C
C
C
C
C
C
C
        SUBROUTINE ROWOP(MAT,RC)
          REAL MAT(3,6)
          INTEGER RC,I,J
          REAL FACTOR
C
C       THIS ROUTINE UNITIZES ONE ROW OF A 3 BY 6 MATRIX (MAT)
C       ABOUT THE PIVOT (RC) AND THEN PERFORMS ELEMENTARY ROW
C       OPERATIONS ON THE REMAINING TWO ROWS.
C       (TO BE USED BY SUBROUTINE INVERT)
C
        FACTOR=MAT(RC,RC)
        DO 100 I=1,6
          MAT(RC,I)=MAT(RC,I)/FACTOR
100     CONTINUE
```

```
      DO 200 J=1,3
        IF (J .EQ. RC) GOTO 200
        IF (ABS(MAT(J,RC)) .LE. .0000000001) GOTO 200
        FACTOR=-MAT(J,RC)/MAT(RC,RC)
        DO 400 I=1,6
          MAT(J,I)=MAT(J,I)+MAT(RC,I)*FACTOR
400     CONTINUE
200   CONTINUE
      END
C
C
C
C
      SUBROUTINE INVERT(MOI,IMOI)
C
C     THIS ROUTINE SETS UP A 3 BY 6 MATRIX (DUMMY).  LEFT 3 BY 3
C     MATRIX IS THE MOMENT OF INERTIA MATRIX (MOI).  THE RIGHT
C     3 BY 3 IS THE IDENTITY MATRIX.  CALLS TO SUBROUTINE ROWOP
C     PERFORM ELEMENTARY ROW OPERATIONS MAKING THE LEFT 3 BY 3
C     OF DUMMY THE IDENTITY MATRIX AND THE RIGHT 3 BY 3 THE INVERSE
C     MOMENT OF INERTIA MATRIX (IMOI).
C
      REAL MOI(3,3),IMOI(3,3),DUMMY(3,6)
      INTEGER K,L
C
      DO 100 K=1,3
        DO 300 L=1,3
          DUMMY(K,L)=MOI(K,L)
          DUMMY(K,L+3)=0.0
300     CONTINUE
        DUMMY(K,K+3)=1.0
100   CONTINUE
      CALL ROWOP(DUMMY,1)
      CALL ROWOP(DUMMY,2)
      CALL ROWOP(DUMMY,3)
      DO 150 K=1,3
        DO 350 L=1,3
          IMOI(K,L)=DUMMY(K,L+3)
350     CONTINUE
150   CONTINUE
      END
C
C
C
C
      SUBROUTINE JETSEL(IMOI,JET,MASS,THRUST,VUN,PUN,PZIUN,
     *  PZHI,TABLE)
C
C     SUMS FORCES AND TORQUES OF THRUSTERS, CONVERTS TO ACCELERATIONS
C     ASSIGNS PROPELLANT LOSSES FOR THRUSTER.
C
      INTEGER I,J,TABLE(4,12,9),ROW,COL
      REAL M(12),ALPHA(3),IMOI(3,3),JET(44,12),MASS,THRUST(44,9)
      REAL VUN(12),PUN(12),PZIUN(12),PZHI(12)
C
      DO 100 I=1,44
        M(1)=JET(I,1)
        M(2)=JET(I,2)
        M(3)=JET(I,3)
        M(4)=JET(I,8)
```

```
            M(5)=JET(I,9)
            M(6)=JET(I,10)
C
C           ASSIGNING OF PROPELLANT
C
            M(7)=0.0
            M(9)=0.0
            M(8)=JET(I,11)
            IF (JET(I,12) .LE. 1.5) THEN
              M(7)=JET(I,11)
              M(8)=0.0
            ENDIF
            IF (JET(I,12) .GE. 2.5) THEN
              M(9)=JET(I,11)
              M(8)=0.0
            ENDIF
C
C           CONVERT FORCES AND TORQUES TO ACCELERATIONS
C
            M(1)=M(1)/MASS
            M(2)=M(2)/MASS
            M(3)=M(3)/MASS
            DO 400 J=1,3
              ALPHA(J)=IMOI(J,1)*M(4)+IMOI(J,2)*M(5)+IMOI(J,3)*M(6)
400         CONTINUE
            M(4)=ALPHA(1)
            M(5)=ALPHA(2)
            M(6)=ALPHA(3)
C
C           ASSIGN THRUST ARRAY AS FOLLOWS
C           COL 1-3  TRANSLATIONAL ACCELERATION (FPS2)
C           COL 4-6  ROTATIONAL ACCELERATION (RAD/S2))
C           COL 7-9  FUEL USED (LBS/S) (7-FWD 8-RT 9-LT)
C
            DO 200 J=1,9
              THRUST(I,J)=M(J)
200         CONTINUE
100       CONTINUE
C
C           ASSIGN RATES CORRESPONDING TO COMMANDS FOR GIVEN
C           THRUST PROFILE
C
          DO 500 I=1,12
            VUN(I)=0.0
            DO 600 J=1,9
              IF (TABLE(1,I,J) .EQ. 0) GOTO 500
              ROW=TABLE(1,I,J)
              COL=INT((I+1)/2)
              VUN(I)=VUN(I)+THRUST(ROW,COL)
600         CONTINUE
500       CONTINUE
C
          DO 525 I=1,12
            PUN(I)=0.0
            DO 625 J=1,9
              IF (TABLE(2,I,J) .EQ. 0) GOTO 525
              ROW=TABLE(2,I,J)
              COL=INT((I+1)/2)
              PUN(I)=PUN(I)+THRUST(ROW,COL)
625         CONTINUE
```

```
525   CONTINUE
C
      DO 550 I=1,12
        PZIUN(I)=0.0
        DO 650 J=1,9
          IF (TABLE(3,I,J) .EQ. 0) GOTO 550
          ROW=TABLE(3,I,J)
          COL=INT((I+1)/2)
          PZIUN(I)=PZIUN(I)+THRUST(ROW,COL)
650     CONTINUE
550   CONTINUE
C
      DO 575 I=1,12
        PZHI(I)=0.0
        DO 675 J=1,9
          IF (TABLE(4,I,J) .EQ. 0) GOTO 575
          ROW=TABLE(4,I,J)
          COL=INT((I+1)/2)
          PZHI(I)=PZHI(I)+THRUST(ROW,COL)
675     CONTINUE
575   CONTINUE
C
      END
```

APPENDIX E

MAIN PROGRAM LISTING

```
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C    PRODUCED BY THE UNITED STATES AIR FORCE ACADEMY
C    DEPARTMENT OF ASTRONAUTICS AND IS NOT PROTECTED
C    BY COPYRIGHT.              (17 U.S.C. SECTION 105)
C        PROGRAM IS BASED ON WORK BY USAFA/DFAS.
C    AUTOVON 259-4110           COMMERCIAL 303-472-4110
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
C      EXECUTES THRUST COMMANDS FOR PROXIMITY OPERATIONS
C      DAP PANEL MODE DEFINITIONS CAN BE FOUND IN:
C      FLIGHT PROCEDURES HANDBOOK/PROXIMITY OPERATIONS
C      PRELIMINARY  NOV 11,1982  NASA
C      PAGES  3-16 TO 3-20
C      QUESTIONS ?????  CONTACT CAPT ALFANO, DFAS
C
       INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'
C
       REAL X(6),T(3,3),Q(4),JET(4,12),JETSEL(44,9),RT,RVX
       REAL WB(3),MAXWX,MAXWY,MAXWZ,ALT,AB(3),AR(3),ANG(3)
       REAL FUEL(3),DELTAT,DVX,DVY,DVZ,DWX,DWY,DWZ,GAS(3),FX(6)
       REAL DBX,DBY,DBZ,SNAP(3),REF(4),DELT1,DELT2,INCLIN,TIME
       REAL TW(3,3),TARG(3),AP(4),RANGE,OMEGA,APSTOP,OLDX(3)
       REAL UP(3),HAFWAY,RRATE,TFUEL,FUDGE,STOPIT
C
       INTEGER*4 SYS$SETEF,SYS$WAITFR,CHAN,COUNT
       INTEGER*2 IBUF(337),IOSB(4)
       INTEGER S(24),C(12),P(12),I,J,PICK,L(24),CO(12),COMM,LITE
       INTEGER DC(12),NLOOPS,TABLE(4,12,9),PCOUNT,APSW,RESET,FLAG
       INTEGER PIC,DS(24),DL(24),OVHD(12),IOSTAT,TRKCNT,RESTOP
       CHARACTER*16 TIMBUF
       INTEGER*4 TIMADR
C
C         AB(3) - ACCELERATION (BODY FRAME)
C         ALT - ALTITUDE OF TARGET (KM)
C         ANG(3) - ANGULAR ACCELERATION (BODY FRAME)
C         APSTOP - RANGE FROM TARGET YOU WISH AUTOPILOT TO STOP
C         APSW - AUTOPILOT SWITCH (S(3)) FROM LAST ITERATION
C         AP(4) - UNIT VECTOR FROM TARGET TO SHUTTLE FOR AUTOPILOT
C         AR(3) - ACCELERATION (REF FRAME)
C         CHAN - CHANNEL FOR PHYSICAL I/O WITH PS300
C         COMM - 1=RHC IN NEUTRAL, 0=RHC NOT IN NEUTRAL
C         COUNT - COUNTER FOR FAST AUTOPILOT GRAPHICS
C         DBX - X DEADBAND FOR ATTITUDE CONTROL
C         DBY - Y DEADBAND FOR ATTITUDE CONTROL
C         DBZ - Z DEADBAND FOR ATTITUDE CONTROL
C         DELT1,2 - TIME COUNTERS TO SEE IF LOOP EXCEEDS .04 SEC
C         DELTAT - TIME STEP
C         DVX - X VELOCITY PULSE
C         DVY - Y VELOCITY PULSE
C         DVZ - Z VELOCITY PULSE
C         DWX - X ROTATIONAL PULSE
C         DWY - Y ROTATIONAL PULSE
C         DWZ - Z ROTATIONAL PULSE
C         FLAG - A PARAMETER TO SLOW DOWN TRANSMISSION RATE TO PS300
C         FUDGE - FUDGE FACTOR TO CHANGE STS RESPONSIVENESS
C         FUEL(3) - TOTAL FUEL EXPENDED FROM EACH TANK
C         FX(6) - PREDICTED FUTURE POS/VEL OF SHUTTLE
C         GAS(3) - FUEL CONSUMED FOR PRESENT ITERATION (FROM EACH T...)
C         HAFWAY - HALF THE DISTANCE SINCE LAST AUTOPILOT UPDATE
C                  (USED FOR BRAKING PURPOSES)
```

```
C       INCLIN - INCLINATION OF TARGET ORBIT (DEG)
C       IBUF - BUFFER FOR ALL MOVEMENT (I/O WITH PS300)
C       IOSB - UNKNOWN PARAMETER (?) FOR I/O WITH PS300
C       IOSTAT - FLAG FOR CHECKING I/O ON UNIT 20
C       JET(4,12) - RESPONSE TO COMMANDS
C               (ROW: 1-VERNIER  2-PRIMARY  3-LOW Z  4-HIGH Z)
C       JETSEL(44,9) - PRECOMPUTED ACCEL AND FUEL FOR EACH THRUSTER
C       LITE - 1=LVLH+3 DISC MODES SELECTED, 0=NOT SELECTED
C       MAXWX - DISC RATE FOR ROLL
C       MAXWY - DISC RATE FOR PITCH
C       MAXWZ - DISC RATE FOR YAW
C       NLOOPS - LOOP COUNTER
C       OLDX(3) - ORIGINAL SHUTTLE POSITION FROM TARGET IN C-W FRAME
C       OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH (RAD/SEC)
C       P(12)- PULSE COMMAND COUNTER
C       PCOUNT - A COUNTER FOR PRINT STATEMENTS
C       PIC - FLAG FOR PS300 GRAPHICS USAGE
C       PICK - NUMBER OF SELECTED RESPONSE MATRIX (JET)
C       Q(4) - QUATERNION
C       RANGE - RANGE FROM TARGET
C       REF(4) - QUATERNION (LVLH TO C-W FRAME)
C       RESET - COMMAND FLAG TO START OVER FROM ORIGINAL POSITION
C       RESTOP - COUNTER TO ALLOW STOPPING OF PROGRAM W/ RESET BUTTON
C       RRATE - RANGE RATE FROM TARGET
C       RT - RENDEZVOUS TIME FOR BANANA AUTOPILOT
C       RVX - FINAL RENDEZVOUS OFFSET DISTANCE (35 FT)
C       SNAP(3) - SNAPSHOT ANGLE (NEEDED FOR DISC RATE MODE)
C       STOPIT - MAXIMUM NUMBER OF MINUTES YOU ANTICIPATE RUNNING
C       T(3,3) - TRANSITION MATRIX (BODY TO REFERENCE FRAME)
C       TABLE(4,12,9) - LIST OF THRUSTERS FIRED VS COMMAND
C       TARG(3) - TARGET POSITION WRT AFT BAY WINDOW
C       TIMBUF,TIMADR - USED TO SET ITERATIVE LOOP TO DELTAT
C       TIME - TIME (SEC)
C       TFUEL - TOTAL FUEL USED
C       TW(3,3) - TRANSITION MATRIX (REF TO WINDOW FRAME)
C       TRKCNT - COUNTER FOR WRITING TO UNIT 20
C       UP(3) - UP VECTOR FOR PS300 AND ALIGNMENT SUBROUTINE
C       WB(3) - ROTATION RATE (BODY FRAME)
C       X(6) - SHUTTLE POSITION/VELOCITY FROM TARGET IN C-W FRAME
C
C       ASSIGNMENT OF DAP PANEL SWITCHES, LIGHTS, AND THRUST COMMANDS
C
C       S( ) - SWITCH ARRAY (READ FROM DAP PANEL)
C       L( ) - LIGHT ARRAY (WRITTEN TO DAP PANEL)
C       C( ) - THRUST COMMAND ARRAY
C       OVHD( ) - COMMAND ARRAY FOR OVERHEAD LOS (READ FROM THC/RHC)
C       CO( ) - COMMAND ARRAY FROM LAST ITERATION
C       DC( ) - DUMMY COMMAND ARRAY
C
C       S(1) - SELECT A / PARAMETERS FOR FAR AWAY MANEUVERING (MAN MODE)
C                       / DIRECT AUTOPILOT (AUTO MODE)
C       S(2) - SELECT B / PARAMETERS FOR CLOSE IN MANEUVERING (MAN MODE)
C                       / C-W (BANANA) AUTOPILOT (AUTO MODE)
C       S(3) - AUTOPILOT
C       S(4) - MAN / MANUAL
C       S(5) - NORM RCS JETS
C       S(6) - VERNIER RCS JETS
C       S(7) - DISCRETE RATE ROLL
C       S(8) - DISCRETE RATE PITCH
C       S(9) - DISCRETE RATE YAW
```

```
C          S(10) - ACCEL ROLL
C          S(11) - ACCEL PITCH
C          S(12) - ACCEL YAW
C          S(13) - PULSE ROLL
C          S(14) - PULSE PITCH
C          S(15) - PULSE YAW
C          S(16) - LVLH
C          S(17) - LOW Z  Y
C          S(18) - HIGH Z
C          S(19) - NORM X
C          S(20) - NORM Y
C          S(21) - NORM Z
C          S(22) - PULSE X
C          S(23) - PULSE Y
C          S(24) - PULSE Z
C
C          L( ) - SAME AS S ARRAY
C
C          C(1) - +X
C          C(2) - -X
C          C(3) - +Y
C          C(4) - -Y
C          C(5) - +Z
C          C(6) - -Z
C          C(7) - +ROL
C          C(8) - -ROL
C          C(9) - +PCH
C          C(10) - -PCH
C          C(11) - +YAW
C          C(12) - -YAW
C
C          CO( ) - SAME AS C ARRAY
C          DC( ) - SAME AS C ARRAY
C
C          1=ON  0=OFF
C
C          INITIALIZATIONS
C
        DATA TIMBUF/'0000 00:00:00.12'/
        DATA (S(I),I=1,24)/24*0/
        DATA (L(I),I=1,12)/1,0,0,1,1,0,3*0,3*1/
        DATA (L(I),I=13,24)/6*0,3*1,3*0/
        DATA (C(I),I=1,12)/12*0/
        DATA (CO(I),I=1,12)/12*0/
        DATA (P(I),I=1,12)/12*0/
        DATA (X(I),I=1,6)/6*0/
        DELTAT=.04
        TIME=0.
        FLAG=0
C
C         SET TIMER
C
        CALL SYSSBINTIM(TIMBUF,TIMADR)
C
C         READ IN JETSEL, TABLE AND JET MATRICES
C
        OPEN(UNIT=2,NAME='[ALFANO.DATA]RESPONSES.DAT',
     +       TYPE='OLD',FORM='UNFORMATTED',READONLY)
C
        DO 115 I=1,44
```

```
          READ(2)(JETSEL(I,J),J=1,9)
115     CONTINUE
C
        DO 120 I=1,12
          READ(2)(TABLE(1,I,J),J=1,9)
          READ(2)(TABLE(2,I,J),J=1,9)
          READ(2)(TABLE(3,I,J),J=1,9)
          READ(2)(TABLE(4,I,J),J=1,9)
          READ(2)JET(1,I)
          READ(2)JET(2,I)
          READ(2)JET(3,I)
          READ(2)JET(4,I)
120     CONTINUE
C
        CLOSE(2)
C
C         ECOCHECK DATA THAT WAS READ IN
C
C        CALL ECOCHK(TABLE,JET)
C
C         DETERMINE USER REQUIREMENTS
C
        PRINT *,' DO YOU WISH :'
        PRINT *,' 1 - REAL TIME GRAPHIC SIMULATION'
        PRINT *,' 2 - FAST GRAPHIC AUTOPILOT DEMO'
        PRINT *,' 3 - FAST AUTOPILOT W/O GRAPHICS'
        READ *,PIC
C
C         SET UP AUTOPILOT, IF CHOSEN
C
        IF ((PIC .EQ. 2).OR.(PIC .EQ. 3)) THEN
          L(1)=0
          L(2)=1
          L(3)=1
          L(4)=0
          APSW=1
          APSTOP=35.
        ENDIF
C
C          READ IN FUDGE FACTOR
C
        PRINT *,'BY WHAT FACTOR DO YOU WISH TO MULTIPLY'
        PRINT *,'STS RESPONSIVENESS ?'
        READ *,FUDGE
C
C          APPLY FUDGE FACTOR TO THRUSTER DATA
C
        DO 37 I=1,44
          DO 38 J=1,9
            JETSEL(I,J)=JETSEL(I,J)*FUDGE
38        CONTINUE
37      CONTINUE
C
C          READ IN MAX RUN TIME
C
        STOPIT=1000.
        IF (PIC .EQ. 1) THEN
          PRINT *,' HOW LONG DO YOU ANTICIPATE RUNNING ?'
          PRINT *,' (IF OVER 10 MINUTES PLEASE COORDINATE',
     +            ' WITH SEILER)'
```

```
        READ *,STOPIT
      ENDIF
C
C       READ IN TARGET ALTITUDE
C
      PRINT *,' ENTER ALTITUDE OF TARGET (KM)'
      READ *,ALT
C
C       COMPUTE ROTATION RATE OF TARGET ABOUT EARTH
C
      OMEGA=SQRT(398600.8/(ALT+6378.135)**3)
C
C       READ IN INCLINATION OF TARGET ORBIT
C
      PRINT *,' ENTER INCLINATION OF TARGET ORBIT (DEG)'
      READ *,INCLIN
C
C       READ IN INITIAL SHUTTLE POSITION
C
      PRINT *,' ENTER COORDINATES OF SHUTTLE FROM TARGET'
      PRINT *,' X,Y,Z CLOHESSY-WILTSHIRE FRAME (FEET)'
      READ *,OLDX(1),OLDX(2),OLDX(3)
C
C       ZERO VELOCITY AND ANGULAR MOTION ARE ASSUMED FOR SHUTTLE.
C       SHUTTLE ALIGNMENT IS INITIALIZED SUCH THAT TARGET
C       STARTS IN CENTER OF SCREEN (-Z IN BODY FRAME)
C
C       INITIALIZE PS300 DISPLAY
C       INITIALIZE BUFFER FOR PS300 MEMORY ADDRESS
C
      IF ((PIC .EQ. 1).OR.(PIC .EQ. 2)) THEN
        CALL PS300(ALT,INCLIN)
        CALL INITBUF(IBUF,CHAN,IOSB)
      ENDIF
C
C       READ AND WRITE TO SHUTTLE MOCKUP
C
      IF (PIC .EQ. 1) CALL IOBUFF(1,S,L,OVHD,RESET)
C
C       OPEN UNIT 20 FOR WRITING AND SET FORMAT.
C       IF UNABLE, PRINT MESSAGE AND END PROGRAM
C
      OPEN(UNIT=20,FILE='STS.TRK',STATUS='NEW',IOSTAT=ISTAT)
      IF (IOSTAT .NE. 0) THEN
        PRINT *,'CAN NOT OPEN OUTPUT FILE'
        PRINT *,'FILE = STS.TRK'
        PRINT *,'ISTAT =',IOSTAT
        GOTO 999
      ENDIF
444   FORMAT('P',4(3X,F8.1),4(3X,F8.6))
445   FORMAT('L',4(3X,F8.1),4(3X,F8.6))
C
C       SET EVENT FLAG FOR WAIT COMMAND
C
      CALL SYS$SETEF(7)
C
C       RESET BEGINS HERE
C
135   CONTINUE
C
```

```
C       INITIALIZE PARAMETERS
C
        SNAP(1)=0.0
        SNAP(2)=0.0
        SNAP(3)=0.0
        FUEL(1)=0.0
        FUEL(2)=0.0
        FUEL(3)=0.0
        WB(1)=0.0
        WB(2)=0.0
        WB(3)=0.0
        FLAG=0
C
        UP(1)=0.0001
        UP(2)=1.0
        UP(3)=0.0001
C
        X(1)=OLDX(1)
        X(2)=OLDX(2)
        X(3)=OLDX(3)
        X(4)=0.0
        X(5)=0.0
        X(6)=0.0
        RESET=0
C
        DO 175 I=1,12
          C(I)=0
          DC(I)=0
175     CONTINUE
C
        HAFWAY=10000000.
C
C          WRITE INITIAL POSITION, TIME, AND QUATERNION
C          TO UNIT 20
C
        WRITE(20,444),X(1),X(2),X(3),TIME,Q(1),Q(2),Q(3),Q(4)
C
C          ALIGN SHUTTLE SO TARGET IS CENTERED IN WINDOW
C
        CALL ALIGN(X(1),X(2),X(3),T,Q,UP)
C
C          INITIALIZE LVLH REF TO ALIGNMENT QUATERNION
C
        REF(1)=Q(1)
        REF(2)=Q(2)
        REF(3)=Q(3)
        REF(4)=Q(4)
C
C          INITIALIZE COUNTERS
C
        NLOOPS=0
        PCOUNT=999
        DELT1=SECNDS(0.)
        TRKCNT=0
        COUNT=0
C
C          ITERATIVE LOOP BEGINS HERE
C
450     CONTINUE
C
```

```
C          UPDATE COUNTERS
C
      NLOOPS=NLOOPS+1
      TIME=NLOOPS*DELTAT
      PCOUNT=PCOUNT+1
      TRKCNT=TRKCNT+1
      COUNT=COUNT+1
      FLAG=FLAG+1
C
C          PICK APPROPRIATE THRUST RESPONSES.
C          VALUES OF PICK ARE AS FOLLOWS:
C          1 - VERNIER
C          2 - PRIMARY
C          3 - PRIMARY WITH +Z INHIBITED
C          4 - PRIMARY WITH HIGH Z
C
C          CHOOSE THRUST RESPONSES FROM DAP PANEL LIGHTS
C
      IF (L(6) .EQ. 1) PICK=1
      IF (L(5) .EQ. 1) PICK=2
      IF (L(17) .EQ. 1) PICK=3
      IF (L(18) .EQ. 1) PICK=4
C
C          SELECT RENDEZVOUS CONSTRAINTS (FROM DAP PANEL LIGHTS)
C
      IF ((L(2)+L(3)) .GT. 0) THEN
        DVX=.01
        DVY=.01
        DVZ=.01
        DWX=.000175
        DWY=.000175
        DWZ=.000175
        DBX=.02
        DBY=.02
        DBZ=.02
        MAXWX=.06
        MAXWY=.06
        MAXWZ=.06
      ELSE
        DVX=.05
        DVY=.05
        DVZ=.05
        DWX=.000873
        DWY=.000873
        DWZ=.000873
        DBX=.1
        DBY=.1
        DBZ=.1
        MAXWX=.3
        MAXWY=.3
        MAXWZ=.3
      ENDIF
C
C          CHECK PULSE MODE LIGHTS
C
      IF (L(22) .EQ. 1) CALL PULSE(P(1),P(2),C(1),C(2),DVX,
     *  DELTAT*JET(PICK,1),DELTAT*JET(PICK,2))
      IF (L(23) .EQ. 1) CALL PULSE(P(3),P(4),C(3),C(4),DVY,
     *  DELTAT*JET(PICK,3),DELTAT*JET(PICK,4))
      IF (L(24) .EQ. 1) CALL PULSE(P(5),P(6),C(5),C(6),DVZ,
```

```
     *  DELTAT*JET(PICK,5),DELTAT*JET(PICK,6))
        IF (L(13) .EQ. 1) CALL PULSE(P(7),P(8),C(7),C(8),DWX,
     *  DELTAT*JET(PICK,7),DELTAT*JET(PICK,8))
        IF (L(14) .EQ. 1) CALL PULSE(P(9),P(10),C(9),C(10),DWY,
     *  DELTAT*JET(PICK,9),DELTAT*JET(PICK,10))
        IF (L(15) .EQ. 1) CALL PULSE(P(11),P(12),C(11),C(12),DWZ,
     *  DELTAT*JET(PICK,11),DELTAT*JET(PICK,12))
C
C          CHECK LVLH MODE USING COMM AND LITE
C
        COMM=0
        LITE=0
        IF (((C(7)+C(8)+C(9)+C(10)+C(11)+C(12)) .EQ. 0) COMM=1
        IF ((L(16)+L(7)+L(8)+L(9)) .EQ. 4) LITE=1
        IF ((LITE+COMM) .EQ. 2) THEN
          CALL LVLH(C,CO,Q,REF,DBX,DBY,DBZ,
     *  WB,JET,DELTAT,MAXWX,MAXWY,MAXWZ,PICK,T)
        ENDIF
C
C          CHECK DISC MODE LIGHTS
C
        IF (L(16) .EQ. 0) THEN
          IF (L(7) .EQ. 1) CALL DISC(JET(PICK,7),JET(PICK,8),WB(1),
     *  DELTAT,MAXWX,C(7),C(8),CO(7),CO(8),DBX,SNAP(1))
          IF (L(8) .EQ. 1) CALL DISC(JET(PICK,9),JET(PICK,10),WB(2),
     *  DELTAT,MAXWY,C(9),C(10),CO(9),CO(10),DBY,SNAP(2))
          IF (L(9) .EQ. 1) CALL DISC(JET(PICK,11),JET(PICK,12),WB(3),
     *  DELTAT,MAXWZ,C(11),C(12),CO(11),CO(12),DBZ,SNAP(3))
        ENDIF
C
C          CHECK IF THE AUTOPILOT (A OR B) IS SELECTED
C             A - DIRECT APPROACH
C             B - CLOSED LOOP CLOHESSY-WILTSHIRE
C
        IF (L(3) .EQ. 1) THEN
C
C           ZERO OUT THC COMMANDS
C
          C(1)=0
          C(2)=0
          C(3)=0
          C(4)=0
          C(5)=0
          C(6)=0
C
C           PREDICT POS/VEL PRODUCED BY RHC COMMANDS AND DRIFT
C
          FX(1)=X(1)
          FX(2)=X(2)
          FX(3)=X(3)
          FX(4)=X(4)
          FX(5)=X(5)
          FX(6)=X(6)
C
          CALL THRUST(JETSEL,AB,ANG,GAS,C,PICK,TABLE)
          CALL BTOR(T,AB,AR)
          CALL LINTEG(FX,OMEGA,AR,DELTAT)
C
C           CHOOSE PROPER AUTOPILOT
C
```

```
            IF (L(1) .EQ. 1) THEN
              CALL APILOT(L,C,REF,X,FX,T,APSW,
     *          S(3),AP,DELTAT,Q,APSTOP,JET,PICK)
             ELSE
              CALL BANANA(FX,OMEGA,C,T,REF,L,
     *          S(3),APSW,Q,HAFWAY,RT,DELTAT,UP,JET,PICK,RVX)
            ENDIF
          ENDIF
C
C           IF NO OTHER MODE IS CHOSEN, NORM/ACCEL MODE IS ASSUMED.
C           COMMANDS REMAIN THE SAME, SO NO SUBROUTINE IS NEEDED.
C
C           APPLY COMMANDS (C) TO RESPONSE MATRIX (JETSEL)
C
          CALL THRUST(JETSEL,AB,ANG,GAS,C,PICK,TABLE)
C
C           TRANSLATE BODY ACCELERATIONS (AB) TO REF FRAME (AR)
C
          CALL BTOR(T,AB,AR)
C
C           APPLY ACCELERATIONS TO DETERMINE POSITION (X(1-3))
C           AND VELOCITY (X(4-6)) OF SHUTTLE RELATIVE TO TARGET
C           USING CLOHESSY-WILTSHIRE EQUATIONS.
C
          CALL LINTEG(X,OMEGA,AR,DELTAT)
C
C           APPLY ANGULAR ACCELERATIONS TO FIGURE QUATERNIONS AND NEW T MATR
C
          CALL ROTATE(ANG,DELTAT,WB,Q,T)
C
C           UPDATE FUEL USED
C
          DO 900 I=1,3
            FUEL(I)=FUEL(I)+GAS(I)*DELTAT
900       CONTINUE
          TFUEL=FUEL(1)+FUEL(2)+FUEL(3)
C
C           DETERMINE TARGET POSITION AND ATTITUDE
C           AS SEEN THRU AFT BAY WINDOW.
C           FOR REAL SIMULATION, DISPLAY ON THE PS300.
C
          IF (PIC .EQ. 1)
     +      CALL LOOK(T,X,OMEGA,TIME,TFUEL,IBUF,CHAN,IOSB)
C
          IF ((PIC .EQ. 2).AND.(COUNT .EQ. 13)) THEN
            CALL LOOK(T,X,OMEGA,TIME,TFUEL,IBUF,CHAN,IOSB)
            COUNT=0
          ENDIF
C
C           WRITE POSITION, TIME, AND ATTITUDE DATA
C           TO UNIT 20 EVERY 10 SECONDS.
C
          IF (TRKCNT .GE. 250) THEN
            WRITE(20,445),X(1),X(2),X(3),TIME,Q(1),Q(2),Q(3),Q(4)
            TRKCNT=0
          ENDIF
C
C           UPDATE OLD COMMAND ARRAY (CO)
C
          DO 950 I=1,12
```

```
          CO(I)=DC(I)
  950   CONTINUE
  C
  C        UPDATE AUTOPILOT SWITCH
  C
        APSW=S(3)
  C
  C        PRINT TO SCREEN EVERY 1000 ITERATIONS IF PIC=3
  C
        IF ((PCOUNT .GE. 1000) .AND. (PIC .EQ. 3))THEN
           PCOUNT=0
           PRINT *,' RANGE (FT)   RANGE RATE (FPS)',
       *     '  FUEL USED (LBS)  TIME (SEC)'
             RANGE=SQRT(X(1)**2+X(2)**2+X(3)**2)
             RRATE=(X(1)*X(4)+X(2)*X(5)+X(3)*X(6))/RANGE
           PRINT *,RANGE,RRATE,(FUEL(1)+FUEL(2)+FUEL(3)),TIME
        ENDIF
  C
  C        ZERO OUT SWITCHES AND RESTORE THRUST COMMANDS,
  C        SET MANUAL LIGHT FLASHING IF AUTOPILOT ON
  C
        DO 645 I=1,12
          S(I)=0
          S(I+12)=0
          C(I)=DC(I)
  645   CONTINUE
  C
        IF ((L(3).EQ.1).AND.((TIME-INT(TIME)).GT..5)) L(4)=1
  C
  C        READ SWITCHES AND THRUST COMMANDS FROM SHUTTLE MOCKUP.
  C        WRITE TO LIGHTS ON DAP PANEL OF SHUTTLE MOCKUP.
  C        IF RESET BUTTON HELD FOR ONE SECOND, STOP PROGRAM
  C        IF RESET COMMANDED, GO TO 135
  C
        IF ((PIC .EQ. 1).AND.(FLAG .GE. 3)) THEN
          CALL IOBUFF(2,S,L,OVHD,RESET)
          CALL SYS$WAITFR(7)
          CALL SYS$SETIMR(7,TIMADR,,)
          IF (RESET .EQ. 1) THEN
            RESTOP=RESTOP+1
            IF (RESTOP .EQ. 15) GOTO 999
            GOTO 135
           ELSE
            RESTOP=0
          ENDIF
          FLAG=0
  C
  C        CHANGE INPUT TO REFLECT OVERHEAD LINE OF SIGHT
  C
          DC(1)=OVHD(6)
          DC(2)=OVHD(5)
          DC(3)=OVHD(3)
          DC(4)=OVHD(4)
          DC(5)=OVHD(1)
          DC(6)=OVHD(2)
          DC(7)=OVHD(12)
          DC(8)=OVHD(11)
          DC(9)=OVHD(9)
          DC(10)=OVHD(10)
          DC(11)=OVHD(7)
```

```
            DC(12)=OVHD(8)
C
C            ASSIGN REAL COMMAND ARRAY (C) WITH DUMMY COMMANDED VALUES
C            THIS IS TO PRESERVE DATA READ FROM THC/RHC WHILE PROGRAM
C            CHANGES C ARRAY DUE TO VARIOUS SELECTED MODES
C
            DO 475 I=1,12
              C(I)=DC(I)
475         CONTINUE
C
          ENDIF
C
C            IF AUTOPILOT ON, SET L(4) (MANUAL LIGHT) TO ZERO
C
          IF (L(3) .EQ. 1) L(4)=0
C
C            APPLY SWITCH COMMANDS TO DAP PANEL LIGHTS
C
          CALL SWITCH(S,L)
C
C
C            RETURN TO 450 FOR NEXT ITERATION
C
          IF (NLOOPS .LT. (1500*STOPIT)) GOTO 450
C
C            COMPUTE AVERAGE LOOP TIME
C
          DELT2=SECNDS(DELT1)
          PRINT *,' AVERAGE LOOP TIME = ',DELT2/NLOOPS,' SECONDS'
C
C           CLOSE UNIT 20
C
          CLOSE(20)
C
999       CONTINUE
C
          END
```

```
      SUBROUTINE ALIGN(X,Y,Z,T,Q,UP)
C
C     COMPUTES BODY AXIS COMPONENTS IN REFERENCE FRAME.
C     ZBOD IS THE POSITION VECTOR.  XBOD AND YBOD
C     COMPLETE THE RIGHT HANDED SYSTEM.
C     DETERMINES QUATERNION (Q) AND TRANSFORMATION
C     MATRIX (T).
C
      REAL X,Y,Z,T(3,3),Q(4),UP(3),CONST
      REAL XBOD(3),YBOD(3),ZBOD(3)
C
C        CAMERA VECTOR BODY COMPONENTS (0,0,-1)
C
      ZBOD(1)=X
      ZBOD(2)=Y
      ZBOD(3)=Z
C
C      DO NOT ALLOW ZBOD TO LIE DIRECTLY ON AN AXIS.
C      THIS IS DONE TO PREVENT QUATERNION AMBIGUITIES.
C
      IF (ABS(ZBOD(1)) .LT. .001) ZBOD(1)=.001
      IF (ABS(ZBOD(2)) .LT. .001) ZBOD(2)=.001
      IF (ABS(ZBOD(3)) .LT. .001) ZBOD(3)=.001
C
C      COMPUTE TRANSFORMATION MATRIX
C
      CALL UNITIZE(ZBOD)
      CALL CROSS(YBOD,ZBOD,UP)
      CALL UNITIZE(YBOD)
      CALL CROSS(XBOD,YBOD,ZBOD)
      CALL UNITIZE(XBOD)
C
      UP(1)=XBOD(1)
      UP(2)=XBOD(2)
      UP(3)=XBOD(3)
C
      T(1,1)=XBOD(1)
      T(1,2)=YBOD(1)
      T(1,3)=ZBOD(1)
      T(2,1)=XBOD(2)
      T(2,2)=YBOD(2)
      T(2,3)=ZBOD(2)
      T(3,1)=XBOD(3)
      T(3,2)=YBOD(3)
      T(3,3)=ZBOD(3)
C
C        COMPUTE QUATERNIONS
C
      Q(4)=1.0+T(1,1)+T(2,2)+T(3,3)
      IF (Q(4) .LT. .1E-30) Q(4)=.1E-30
      Q(4)=SQRT(Q(4))/2.0
      Q(1)=(T(3,2)-T(2,3))/(4.0*Q(4))
      Q(2)=(T(1,3)-T(3,1))/(4.0*Q(4))
      Q(3)=(T(2,1)-T(1,2))/(4.0*Q(4))
C
C        NORMALIZE QUATERNIONS
C
      CONST=SQRT(Q(1)*Q(1)+Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4))
      Q(1)=Q(1)/CONST
      Q(2)=Q(2)/CONST
```

```
      Q(3)=Q(3)/CONST
      Q(4)=Q(4)/CONST
C
      END
```

```
      SUBROUTINE APILOT(L,C,REF,X,FX,T,OLDSW,NEWSW,
     *  AP,DELTAT,Q,APSTOP,JET,PICK)
C
C     THIS AUTOPILOT FLIES THE SHUTTLE DIRECTLY TO
C     TARGET (DIRECT APPROACH) USING LVLH FOR ATTITUDE
C     CONTROL.  IT 'THINKS' ONE ITERATION IN ADVANCE
C     USING PREDICTED POS/VEL DUE TO RHC THRUSTING
C     AND ORBITAL DRIFT.
C
      REAL REF(4),X(6),MAXD,DELTAV(3),VDB,DELTAT
      REAL AP(4),DP(3),Q(4),APSTOP,FX(6),JET(4,12)
      REAL DELVB(3),T(3,3),RANGE,REQV(3)
      INTEGER L(24),C(12),I,OLDSW,NEWSW,PICK
C
C        REF - REFERENCE QUATERNION FOR ATTITUDE CONTROL
C        X - PRESENT POS AND VEL OF SHUTTLE FROM TARGET
C        FX - FUTURE POS AND VEL OF SHUTTLE FROM TARGET
C        MAXD - MAX DISTANCE CHANGE FOR ONE ITERATION
C        DELTAV - VELOCITY CHANGE (TARGET FRAME)
C        DELTAT - TIME INCREMENT
C        VDB - VELOCITY DEADBAND
C        AP - REFERENCE UNIT VECTOR FROM TARGET TO SHUTTLE
C        DP - DESIRED POSITION
C        DELVB - VELOCITY CHANGE (BODY FRAME)
C        T - TRANSFORMATION MATRIX
C        L - DAP PANEL LIGHTS
C        C - THC/RHC COMMANDS
C        OLDSW - LAST POSITION OF AUTOPILOT SELECT SWITCH
C        NEWSW - PRESENT POSITION OF AUTOPILOT SELECT SWITCH
C        APSTOP - RANGE YOU  WISH TO MAINTAIN FROM TARGET
C        REQV - REQUIRED VELOCITY
C        Q - PRESENT QUATERNION
C
C        COMPUTE RANGE
C
      RANGE=SQRT(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))
C
C        INITIALIZE LIGHTS AND COMMANDS
C
      DO 100 I=1,12
        L(I)=0
        L(I+12)=0
100   CONTINUE
      L(1)=1
      L(3)=1
      L(5)=1
      L(7)=1
      L(8)=1
      L(9)=1
      L(16)=1
      L(20)=1
C
C        COMPUTE MAX DISTANCE CHANGE
C
      MAXD=.005*(RANGE)*DELTAT
      IF (RANGE .LT. 1.5*APSTOP) THEN
        MAXD=MAXD*(RANGE-APSTOP)/(.5*APSTOP)
      ENDIF
C
C        SWITCH TO LOW Z INSIDE 200 FEET
```

```
C
      IF (RANGE .GE. 200.0) THEN
        L(21)=1
      ELSE
        L(17)=1
      ENDIF
C
C         CHECK TO SEE IF AUTOPILOT JUST SELECTED
C
      IF (OLDSW .NE. NEWSW) THEN
C
C         SET REF TO PRESENT QUATERNION
C         (TO BE USED BY LVLH)
C
        REF(1)=Q(1)
        REF(2)=Q(2)
        REF(3)=Q(3)
        REF(4)=Q(4)
C
C         SET REFERENCE VECTOR
C
        AP(1)=X(1)/RANGE
        AP(2)=X(2)/RANGE
        AP(3)=X(3)/RANGE
        AP(4)=RANGE+1.0
C
      ENDIF
C
C         FIGURE DESIRED FUTUNE POSITION (DP) (+2 ITERATIONS)
C
      IF (ABS(RANGE-APSTOP) .LT. ABS(AP(4)-APSTOP)) THEN
        AP(4)=RANGE
       ELSE
        MAXD=0.0
      ENDIF
C
      DP(1)=(AP(4)-2.0*MAXD)*AP(1)
      DP(2)=(AP(4)-2.0*MAXD)*AP(2)
      DP(3)=(AP(4)-2.0*MAXD)*AP(3)
C
C         COMPUTE REQUIRED VELOCITY (NEXT ITERATION)
C
      REQV(1)=(DP(1)-FX(1))/DELTAT
      REQV(2)=(DP(2)-FX(2))/DELTAT
      REQV(3)=(DP(3)-FX(3))/DELTAT
C
C         FIGURE VELOCITY TO BE GAINED (DELTAV, TARGET FRAME)
C
      DELTAV(1)=REQV(1)-FX(4)
      DELTAV(2)=REQV(2)-FX(5)
      DELTAV(3)=REQV(3)-FX(6)
C
C         TRANSFORM TO BODY FRAME
C
      CALL RTOB(T,DELTAV,DELVB)
C
C         DETERMINE IF THRUST IS REQUIRED
C         (VDB IS VELOCITY DEADBAND)
C
      VDB=1.1*DELTAT
```

```
C
      IF (DELVB(1) .GT. VDB*JET(PICK,1)) C(1)=1
      IF (DELVB(1) .LT. VDB*JET(PICK,2)) C(2)=1
      IF (DELVB(2) .GT. VDB*JET(PICK,3)) C(3)=1
      IF (DELVB(2) .LT. VDB*JET(PICK,4)) C(4)=1
      IF (DELVB(3) .GT. VDB*JET(PICK,5)) C(5)=1
      IF (DELVB(3) .LT. VDB*JET(PICK,6)) C(6)=1
C
      END
```

```
      SUBROUTINE ARM
C
C       THIS ROUTINE BUILDS THE PAYLOAD ARM ON THE PS300
C       ALL DIMENSIONS ARE IN FEET.
C
      INCLUDE 'PROCONST.FOR/NOLIST'
C
      REAL*4 V(3)
C
C   COMMANDS FOR WRIST ROLL
C
C         SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
      CALL PFN('WRROLL','YROTATE',ERR)
C
      CALL PSNST('WR ROLL',1,'DLABEL1',ERR)
      CALL PCONN('DIALS',1,1,'WRROLL',ERR)
C
C         CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC1','ACCUMULATE',ERR)
C
      CALL PCONN('WRROLL',1,1,'ACC1',ERR)
      CALL PSNREA(0.0,2,'ACC1',ERR)
      CALL PSNREA(0.1,3,'ACC1',ERR)
      CALL PSNREA(1.0,4,'ACC1',ERR)
      CALL PSNREA(447.0,5,'ACC1',ERR)
      CALL PSNREA(-447.0,6,'ACC1',ERR)
C
C         CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
      CALL PCONN('ACC1',1,1,'WRIST.ROLL',ERR)
C
C
C   COMMANDS FOR WRIST YAW
C
C         SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
      CALL PFN('WRYAW','ZROTATE',ERR)
C
      CALL PSNST('WR YAW',1,'DLABEL2',ERR)
      CALL PCONN('DIALS',2,1,'WRYAW',ERR)
C
C         CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC2','ACCUMULATE',ERR)
C
      CALL PCONN('WRYAW',1,1,'ACC2',ERR)
      CALL PSNREA(0.0,2,'ACC2',ERR)
      CALL PSNREA(0.1,3,'ACC2',ERR)
      CALL PSNREA(1.0,4,'ACC2',ERR)
      CALL PSNREA(120.0,5,'ACC2',ERR)
      CALL PSNREA(-120.0,6,'ACC2',ERR)
C
C         CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
      CALL PCONN('ACC2',1,1,'WRIST.YAW',ERR)
C
C
C   COMMANDS FOR WRIST PITCH
```

```
C
C          SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
      CALL PFN('WRPITCH','XROTATE',ERR)
C
      CALL PSNST('WR PITCH',1,'DLABEL3',ERR)
      CALL PCONN('DIALS',3,1,'WRPITCH',ERR)
C
C          CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC3','ACCUMULATE',ERR)
C
      CALL PCONN('WRPITCH',1,1,'ACC3',ERR)
      CALL PSNREA(0.0,2,'ACC3',ERR)
      CALL PSNREA(0.1,3,'ACC3',ERR)
      CALL PSNREA(1.0,4,'ACC3',ERR)
      CALL PSNREA(120.0,5,'ACC3',ERR)
      CALL PSNREA(-120.0,6,'ACC3',ERR)
C
C          CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
      CALL PCONN('ACC3',1,1,'WRIST.PITCH',ERR)
C
C
C   COMMANDS FOR ELBOW PITCH
C
C          SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
      CALL PFN('ELPITCH','XROTATE',ERR)
C
      CALL PSNST('EL PITCH',1,'DLABEL4',ERR)
      CALL PCONN('DIALS',4,1,'ELPITCH',ERR)
C
C          CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC4','ACCUMULATE',ERR)
C
      CALL PCONN('ELPITCH',1,1,'ACC4',ERR)
      CALL PSNREA(0.0,2,'ACC4',ERR)
      CALL PSNREA(0.1,3,'ACC4',ERR)
      CALL PSNREA(1.0,4,'ACC4',ERR)
      CALL PSNREA(160.0,5,'ACC4',ERR)
      CALL PSNREA(-2.0,6,'ACC4',ERR)
C
C          CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
      CALL PCONN('ACC4',1,1,'ELBOW.PITCH',ERR)
C
C   COMMANDS FOR SHOULDER YAW
C
C          SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
      CALL PFN('SHYAW','ZROTATE',ERR)
C
      CALL PSNST('SH YAW',1,'DLABEL5',ERR)
      CALL PCONN('DIALS',5,1,'SHYAW',ERR)
C
C          CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC5','ACCUMULATE',ERR)
```

```
C
      CALL PCONN('SHYAW',1,1,'ACC5',ERR)
      CALL PSNREA(0.0,2,'ACC5',ERR)
      CALL PSNREA(0.1,3,'ACC5',ERR)
      CALL PSNREA(1.0,4,'ACC5',ERR)
      CALL PSNREA(180.0,5,'ACC5',ERR)
      CALL PSNREA(-180.0,6,'ACC5',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
      CALL PCONN('ACC5',1,1,'SHOULDER.YAW',ERR)
C
C  COMMANDS FOR SHOULDER PITCH
C
C        SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
      CALL PFN('SHPITCH','XROTATE',ERR)
C
      CALL PSNST('SH PITCH',1,'DLABEL6',ERR)
      CALL PCONN('DIALS',6,1,'SHPITCH',ERR)
C
C        CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC6','ACCUMULATE',ERR)
C
      CALL PCONN('SHPITCH',1,1,'ACC6',ERR)
      CALL PSNREA(0.0,2,'ACC6',ERR)
      CALL PSNREA(0.1,3,'ACC6',ERR)
      CALL PSNREA(1.0,4,'ACC6',ERR)
      CALL PSNREA(2.0,5,'ACC6',ERR)
      CALL PSNREA(-145.0,6,'ACC6',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
      CALL PCONN('ACC6',1,1,'SHOULDER.PITCH',ERR)
C
C        CREATE A 3D VECTOR (IN FEET) TO TRANSLATE ARM
C
      CALL PFN('XYVEC','VEC',ERR)
      CALL PFN('XYZVEC','VEC',ERR)
      CALL PCONN('XYVEC',1,1,'XYZVEC',ERR)
      CALL PCONN('XYZVEC',1,1,'SHOULDER.TRAN',ERR)
C
C
C  BUILD A WRIST
C
      CALL PBEGS('WRIST',ERR)
        CALL PROTX('PITCH',0.0,'"',ERR)
          V(1)=.5415
          V(2)=1.23
          V(3)=.5415
        CALL PSCALE('"',V,'CYLINDER',ERR)
          V(1)=0.0
          V(2)=1.23
          V(3)=0.0
        CALL PTRANS('"',V,'"',ERR)
        CALL PROTZ('YAW',0.0,'"',ERR)
        CALL PROTY('ROLL',0.0,'"',ERR)
          V(1)=.5415
          V(2)=4.93
```

```
            V(3)=.5415
          CALL PSCALE('"',V,'CYLINDER',ERR)
        CALL PENDS(ERR)
C
C   BUILD A FOREARM (W/ ELBOW)
C
        CALL PBEGS('ELBOW',ERR)
            V(1)=0.0
            V(2)=0.0
            V(3)=.5415
          CALL PTRANS('"',V,'"',ERR)
          CALL PROTX('PITCH',0.0,'"',ERR)
            V(1)=0.0
            V(2)=0.0
            V(3)=-.5415
          CALL PTRANS('"',V,'"',ERR)
            V(1)=.5415
            V(2)=23.1625
            V(3)=.5415
          CALL PSCALE('"',V,'CYLINDER',ERR)
            V(1)=0.0
            V(2)=23.1625
            V(3)=0.0
          CALL PTRANS('"',V,'WRIST',ERR)
        CALL PENDS(ERR)
C
C   BUILD THE SHOULDER
C
        CALL PBEGS('SHOULDER',ERR)
            V(1)=0.0
            V(2)=0.0
            V(3)=0.0
          CALL PTRANS('TRAN',V,'"',ERR)
          CALL PROTZ('"',90.0,'CYL',ERR)
            V(1)=.58
            V(2)=1.083
            V(3)=.58
          CALL PSCALE('CYL',V,'CYLINDER',ERR)
          CALL PROTX('"',90.0,'"',ERR)
          CALL PROTX('PITCH',0.0,'"',ERR)
          CALL PROTZ('YAW',0.0,'"',ERR)
            V(1)=.5415
            V(2)=20.921
            V(3)=.5415
          CALL PSCALE('"',V,'CYLINDER',ERR)
            V(1)=0.C
            V(2)=20.921
            V(3)=0.0
          CALL PTRANS('"',V,'ELBOW',ERR)
        CALL PENDS(ERR)
C
C RESCALE SHOULDER FROM FEET TO KM AND NAME IT 'ARM'
C
            V(1)=.0003048
            V(2)=.0003048
            V(3)=.0003048
        CALL PSCALE('ARM',V,'SHOULDER',ERR)
C
        END
```

```
      SUBROUTINE BANANA(X,OMEGA,C,T,REF,L,
     *NEWSW,OLDSW,Q,HAFWAY,RT,DELTAT,UP,JET,PICK,RVX)
C
C     THIS IS AN EXPLICIT GUIDANCE SCHEME.
C     IT SOLVES THE CLOHESSY-WILTSHIRE EQUATIONS FOR
C     VELOCITY REQUIRED (VX,VY,VZ, TARGET FRAME) GIVEN
C     A SPECIFIED RENDEZVOUS TIME (RT).  VELOCITY TO
C     BE GAINED (DELTAV) IS THEN COMPUTED AND TRANSFORMED
C     TO BODY FRAME (DELVB).  IF OUTSIDE THE VELOCITY
C     DEADBAND (VDB), THRUST IS COMMANDED.
C     LVLH IS USED TO  MAINTAIN ATTITUDE WITH REPEATED
C     CALLS TO 'ALIGN'.
C
      REAL X(6),OMEGA,REF(4),DUMT(3,3),VX,VY
      REAL T(3,3),DELTAV(3),DELVB(3),A,D,E,F
      REAL RANGE,B,VZ,Q(4),HAFWAY,RT,DELTAT,UP(3)
      REAL POS(3),BPOS(3),JET(4,12),RVX
C
      INTEGER C(12),I,L(24),NEWSW,OLDSW,PICK
C
C        L - DAP PANEL LIGHTS
C        T - TRANSFORMATION MATRIX
C        DUMT - DUMMY TRANSFORMATION MATRIX
C        REF - REFERENCE QUATERNION FOR LVLH MODE
C        OMEGA - ROTATION RATE OF TARGET ABOUT EARTH
C        C - COMMAND SWITCHES
C        DELTAV - VEL CHANGE (TARGET FRAME)
C        DELVB - VEL CHANGE (BOD FRAME)
C        RANGE - RANGE FROM TARGET
C        RT - RENDEZVOUS TIME (SECONDS)
C        A,B,D,E,F - INTERMEDIATE VARIABLES
C        VX,VY,VZ - VELOCITY NEEDED
C        NEWSW - PRESENT CONDITION OF S(3)
C        OLDSW - LAST CONDITION OF S(3)
C        HAFWAY - HALF DISTANCE TO TARGET
C        X(6) - FUTURE PREDICTED POS/VEL
C        DELTAT - TIME INCREMENT
C        UP - UP VECTOR (FOR ALIGNMENT)
C        JET - ACCELERATION AVAILABLE (F/S**2)
C        PICK - THRUST PROFILE SELECTED
C        RVX - FINAL RENDEZVOUS OFFSET DISTANCE
C
C        INITIALIZATIONS
C
      RANGE=SQRT(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))
C
      DO 100 I=1,12
        L(I)=0
        L(I+12)=0
100   CONTINUE
C
C        SET DAP PANEL LIGHTS
C
      L(2)=1
      L(3)=1
      L(5)=1
      L(7)=1
      L(8)=1
      L(9)=1
      L(16)=1
```

```
      L(20)=1
C
C        USE LOW Z FROM 60-200 FEET
C
      IF ((RANGE.LE.200.0).AND.(RANGE.GE.60.)) THEN
        L(17)=1
      ELSE
        L(21)=1
      ENDIF
C
C       UPDATE TIME TO RENDEZVOUS
C
      RT=RT-DELTAT
C
C       CHECK TO SEE IF AUTOPILOT JUST SELECTED
C
      IF (OLDSW .NE. NEWSW) THEN
        HAFWAY=1000000.
        RVX=0.
      ENDIF
C
C       CHECK TO SEE IF YOU ARE 'HALFWAY' THERE
C       (ACTUALLY 4/5 OF THE WAY THERE)
C
      IF (RANGE .LT. HAFWAY) THEN
        RT=4.*RANGE
        IF (RT .GT. 1200.) RT=1200.
        IF (RANGE .GE. 250.) THEN
          HAFWAY=250.
         ELSE
          HAFWAY=(RANGE-35.)/5.+35.
        ENDIF
        IF (X(1) .LT. 0.0) THEN
          RVX=+35.
         ELSE
          RVX=-35.
        ENDIF
      ENDIF
C
      IF (RT .LT. 0.0) RT=100.
C
C       COMPUTE NEW REFERENCE QUATERNION BASED ON POSITION
C
      CALL ALIGN(X(1),X(2),X(3),DUMT,REF,UP)
C
C          CHANGE X(1) TO MISS HITTING TARGET BY
C          RVX DISANCE FEET
C
      X(1)=X(1)+RVX
C
C          COMPUTE DELTAV (TARGET FRAME)
C
      SW=SIN(OMEGA*RT)
      CW=COS(OMEGA*RT)
C
      A=-3.0*RT+4.0*SW/OMEGA
      B=(2.0/OMEGA)*(CW-1.0)
      F=X(1)-6.0*OMEGA*X(2)*RT+6.0*X(2)*SW
      D=SW/OMEGA
      E=4.0*X(2)-3.0*X(2)*CW
```

```
C
      VY=(-E-F*B/A)/(B*B/A+D)
      VX=(-B*VY-F)/A
      VZ=-X(3)*OMEGA*CW/SW
C
      DELTAV(1)=VX-X(4)
      DELTAV(2)=VY-X(5)
      DELTAV(3)=VZ-X(6)
C
C        TRANSFORM TO BODY FRAME
C
      CALL RTOB(T,DELTAV,DELVB)
C
C        DETERMINE IF THRUST IS REQUIRED
C        VDB IS VELOCITY DEADBAND
C
      VDB=1.1*DELTAT
C
      IF (DELVB(1) .GT. VDB*JET(PICK,1)) C(1)=1
      IF (DELVB(1) .LT. VDB*JET(PICK,2)) C(2)=1
      IF (DELVB(2) .GT. VDB*JET(PICK,3)) C(3)=1
      IF (DELVB(2) .LT. VDB*JET(PICK,4)) C(4)=1
      IF (DELVB(3) .GT. VDB*JET(PICK,5)) C(5)=1
      IF (DELVB(3) .LT. VDB*JET(PICK,6)) C(6)=1
C
      END
```

```
      SUBROUTINE BTOR(T,BOD,REF)
C
C     TRANSFORMS FROM BOD TO REF FRAME GIVEN TRANSFORMATION MATRIX T
C
      REAL T(3,3),BOD(3),REF(3)
C
      REF(1)=BOD(1)*T(1,1)+BOD(2)*T(1,2)+BOD(3)*T(1,3)
      REF(2)=BOD(1)*T(2,1)+BOD(2)*T(2,2)+BOD(3)*T(2,3)
      REF(3)=BOD(1)*T(3,1)+BOD(2)*T(3,2)+BOD(3)*T(3,3)
C
      END
```

```
      SUBROUTINE CHECK(ANG,DB,WB,C1,C2,JET1,JET2,MAX,DELTAT)
C
C        THIS SUBROUTINE EXAMINES ANGLE (ANG) AND BODY RATE (WB)
C        TO DETERMINE THRUST REQUIRED TO RESTORE ANGLE WITHIN
C        DEADBAND (DB) WHILE NOT EXCEEDING MAX BODY RATE (MAX)
C        A WINDOW IS SET UP (SA) TO STOP ROTATION.  IF INSIDE WINDOW
C        AND APPROACHING ZERO, THRUST TO STOP.
C
      REAL ANG,DB,WB,JET1,JET2,MAX,DELTAT,SA
      INTEGER C1,C2,N
C
C        DELTAT - TIME INCREMENT
C        SA - STOP ANGLE (ANGLE REQUIRED TO MAKE WB=0)
C        N - COUNTER FOR STOP ANGLE SOLUTION
C        JET - ANGULAR ACCELERATION AVAILABLE
C
C        INITIALIZATIONS
C
      C1=0
      C2=0
C
C        CHECK IF OUTSIDE DEADBAND
C
      IF (ABS(ANG) .GT. DB) THEN
C
C        CHECK IF WB IS GOING AWAY FROM SNAPSHOT AND CHOOSE THRUST
C
        IF (ANG*WB .GE. 0.0) THEN
          IF (ANG .GT. 0.0) C2=1
          IF (ANG .LT. 0.0) C1=1
        ELSE
C
C            ELSE WB IS MOVING TOWARDS SNAPSHOT. DETERMINE STOP
C            ANGLE (SA) AND CHOOSE THRUST.
C
          IF (ANG .GT. 0.0) THEN
            SA=ABS(.5*WB*WB/JET1)+DB*.9
C             N=INT(-WB/(JET2*DELTAT))
C             SA=ABS(N*WB+JET2*DELTAT*N*(N-1)/2.0)*DELTAT+.9*DB
            IF (ANG .LE. SA) C1=1
            SA=ABS(.5*(WB+JET2*DELTAT)**2/JET1)
              SA=SA+.9*DB+WB*DELTAT
            IF (ANG .GT. SA) C2=1
          ELSE
            SA=ABS(.5*WB*WB/JET2)+DB*.9
C             N=INT(-WB/(JET1*DELTAT))
C             SA=ABS(N*WB+JET1*DELTAT*N*(N-1)/2.0)*DELTAT+.9*DB
            IF (ABS(ANG) .LE. SA) C2=1
            SA=ABS(.5*(WB+JET1*DELTAT)**2/JET2)
              SA=SA+.9*DB+WB*DELTAT
            IF (ABS(ANG) .GT. SA) C1=1
          ENDIF
        ENDIF
      ENDIF
C
C       COMPARE RATES TO MAX RATE AND SET THRUST COMMANDS
C
      IF (WB .GT. 0.0) THEN
        IF ((WB+DELTAT*JET1) .GT. MAX) C1=0
        IF (WB .GT. MAX) C2=1
```

```
      ELSE
        IF ((WB+DELTAT*JET2) .LT. -MAX) C2=0
        IF (WB .LT. -MAX) C1=1
      ENDIF
C
      END
```

```
      SUBROUTINE CROSS(A,B,C)
C
C     COMPUTES  A = B X C
C
      REAL A(3),B(3),C(3)
C
      A(1)=B(2)*C(3)-B(3)*C(2)
      A(2)=B(3)*C(1)-B(1)*C(3)
      A(3)=B(1)*C(2)-B(2)*C(1)
C
      END
```

```
      SUBROUTINE DISC(JET1,JET2,WB,DELTAT,MAX,C1,C2,CO1,CO2,DB,ANG)
C
C     ATTITUDE CONTROL SCHEME FOR ONE BODY AXIS.  ALSO
C     APPLIES DISCRETE RATE (MAX) TO PRESENT BODY RATE (WB).
C     IF DISC RATE WILL BE (OR IS BEING) EXCEEDED, COMMAND (C1,C2)
C     IS SET TO BRING/MAINTAIN RATE WITHIN LIMITS.
C
      REAL JET1,JET2,DELTAT,MAX,WB,T,DB,SA,ANG
      INTEGER C1,C2,CO1,CO2
C
C       ANG - ANGLE FROM WHEN SNAPSHOT TAKEN
C       SNAP - TRIGGER TO TAKE NEW SNAPSHOT
C       CO1,2 - OLD COMMAND FROM RHC
C       C1,2 - PRESENT COMMAND FROM RHC
C       DB - DEADBAND WIDTH (+/-)
C       JET1,2 - ANGULAR ACCELERATION AVAILABLE
C       WB - BODY RATE (FOR SELECTED AXIS)
C
C       COMPUTE PRESENT ANGLE FROM SNAPSHOT ANGLE
C
      ANG=ANG+WB*DELTAT
C
C       TAKE SNAPSHOT (SNAP=1) IF CONTROLLER HAS JUST BEEN MOVED TO NEUTRAL
C
      SNAP=0
      IF ((C1+C2) .EQ. 0) THEN
        IF ((CO1-C1) .GT. 0) SNAP=1
        IF ((CO2-C2) .GT. 0) SNAP=1
        IF (SNAP .EQ. 1) ANG=0.0
C
C          DETERMINE THRUST PROFILE
C
        CALL CHECK(ANG,DB,WB,C1,C2,JET1,JET2,
     *    MAX,DELTAT)
C
      ELSE
C
C      COMPARE RATES TO MAX RATE AND SET THRUST COMMANDS
C
        IF (WB .GT. 0.0) THEN
          IF ((WB+DELTAT*JET1) .GT. MAX) C1=0
          IF (WB .GT. MAX) C2=1
        ELSE
          IF ((WB+DELTAT*JET2) .LT. -MAX) C2=0
          IF (WB .LT. -MAX) C1=1
        ENDIF
      ENDIF
      END
```

```
      SUBROUTINE ECOCHK(TABLE,JET)
C
C       ECOCHECK OF TABLE AND JET
C
      REAL JET(4,12)
      INTEGER I,J,TABLE(4,12,9)
C
175   FORMAT(1X,A5,3X,9(1X,I2),5X,F8.5)
C
      PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
      PRINT *,' ECOCHECK OF VERNIER THRUSTER TABLE AND ACCEL'
      PRINT *,' '
      PRINT 175,'  + X',(TABLE(1,1,J),J=1,9),JET(1,1)
      PRINT 175,'  - X',(TABLE(1,2,J),J=1,9),JET(1,2)
      PRINT 175,'  + Y',(TABLE(1,3,J),J=1,9),JET(1,3)
      PRINT *,' '
      PRINT 175,'  - Y',(TABLE(1,4,J),J=1,9),JET(1,4)
      PRINT 175,'  + Z',(TABLE(1,5,J),J=1,9),JET(1,5)
      PRINT 175,'  - Z',(TABLE(1,6,J),J=1,9),JET(1,6)
      PRINT *,' '
      PRINT 175,'+ ROL',(TABLE(1,7,J),J=1,9),JET(1,7)
      PRINT 175,'- RCL',(TABLE(1,8,J),J=1,9),JET(1,8)
      PRINT 175,'+ PCH',(TABLE(1,9,J),J=1,9),JET(1,9)
      PRINT *,' '
      PRINT 175,'- PCH',(TABLE(1,10,J),J=1,9),JET(1,10)
      PRINT 175,'+ YAW',(TABLE(1,11,J),J=1,9),JET(1,11)
      PRINT 175,'- YAW',(TABLE(1,12,J),J=1,9),JET(1,12)
      PRINT *,' '
      PRINT *,' HIT RETURN TO CONTINUE'
      READ *
C
      PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
      PRINT *,' ECOCHECK OF PRIMARY THRUSTER TABLE AND ACCEL'
      PRINT *,' '
      PRINT 175,'  + X',(TABLE(2,1,J),J=1,9),JET(2,1)
      PRINT 175,'  - X',(TABLE(2,2,J),J=1,9),JET(2,2)
      PRINT 175,'  + Y',(TABLE(2,3,J),J=1,9),JET(2,3)
      PRINT *,' '
      PRINT 175,'  - Y',(TABLE(2,4,J),J=1,9),JET(2,4)
      PRINT 175,'  + Z',(TABLE(2,5,J),J=1,9),JET(2,5)
      PRINT 175,'  - Z',(TABLE(2,6,J),J=1,9),JET(2,6)
      PRINT *,' '
      PRINT 175,'+ ROL',(TABLE(2,7,J),J=1,9),JET(2,7)
      PRINT 175,'- RCL',(TABLE(2,8,J),J=1,9),JET(2,8)
      PRINT 175,'+ PCH',(TABLE(2,9,J),J=1,9),JET(2,9)
      PRINT *,' '
      PRINT 175,'- PCH',(TABLE(2,10,J),J=1,9),JET(2,10)
      PRINT 175,'+ YAW',(TABLE(2,11,J),J=1,9),JET(2,11)
      PRINT 175,'- YAW',(TABLE(2,12,J),J=1,9),JET(2,12)
      PRINT *,' '
      PRINT *,' HIT RETURN TO CONTINUE'
      READ *
C
      PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
      PRINT *,' ECOCHECK OF PZI THRUSTER TABLE AND ACCEL'
      PRINT *,' '
      PRINT 175,'  + X',(TABLE(3,1,J),J=1,9),JET(3,1)
      PRINT 175,'  - X',(TABLE(3,2,J),J=1,9),JET(3,2)
      PRINT 175,'  + Y',(TABLE(3,3,J),J=1,9),JET(3,3)
      PRINT *,' '
```

```
      PRINT 175,'   - Y',(TABLE(3,4,J),J=1,9),JET(3,4)
      PRINT 175,'   + Z',(TABLE(3,5,J),J=1,9),JET(3,5)
      PRINT 175,'   - Z',(TABLE(3,6,J),J=1,9),JET(3,6)
      PRINT *,' '
      PRINT 175,'+ ROL',(TABLE(3,7,J),J=1,9),JET(3,7)
      PRINT 175,'- ROL',(TABLE(3,8,J),J=1,9),JET(3,8)
      PRINT 175,'+ PCH',(TABLE(3,9,J),J=1,9),JET(3,9)
      PRINT *,' '
      PRINT 175,'- PCH',(TABLE(3,10,J),J=1,9),JET(3,10)
      PRINT 175,'+ YAW',(TABLE(3,11,J),J=1,9),JET(3,11)
      PRINT 175,'- YAW',(TABLE(3,12,J),J=1,9),JET(3,12)
      PRINT *,' '
      PRINT *,' HIT RETURN TO CONTINUE'
      READ *
C
      PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
      PRINT *,' ECOCHECK OF PZHI THRUSTER TABLE AND ACCEL'
      PRINT *,' '
      PRINT 175,'   + X',(TABLE(4,1,J),J=1,9),JET(4,1)
      PRINT 175,'   - X',(TABLE(4,2,J),J=1,9),JET(4,2)
      PRINT 175,'   + Y',(TABLE(4,3,J),J=1,9),JET(4,3)
      PRINT *,' '
      PRINT 175,'   - Y',(TABLE(4,4,J),J=1,9),JET(4,4)
      PRINT 175,'   + Z',(TABLE(4,5,J),J=1,9),JET(4,5)
      PRINT 175,'   - Z',(TABLE(4,6,J),J=1,9),JET(4,6)
      PRINT *,' '
      PRINT 175,'+ ROL',(TABLE(4,7,J),J=1,9),JET(4,7)
      PRINT 175,'- ROL',(TABLE(4,8,J),J=1,9),JET(4,8)
      PRINT 175,'+ PCH',(TABLE(4,9,J),J=1,9),JET(4,9)
      PRINT *,' '
      PRINT 175,'- PCH',(TABLE(4,10,J),J=1,9),JET(4,10)
      PRINT 175,'+ YAW',(TABLE(4,11,J),J=1,9),JET(4,11)
      PRINT 175,'- YAW',(TABLE(4,12,J),J=1,9),JET(4,12)
      PRINT *,' '
      PRINT *,' HIT RETURN TO CONTINUE'
      READ *
C
      PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
C
      END
```

```
C
C
C
C     THE FOLLOWING ERROR HANDLER DEMONSTRATES THE GENERAL
C     OVERALL RECOMMENDED FORM THAT THE USER'S OWN ERROR
C     HANDLER SHOULD FOLLOW.
C
C     THIS ERROR HANDLER UPON BEING INVOKED WRITES ALL
C     MESSAGES TO THE DATA FILE:  'PROERROR.LOG'.  ERROR
C     AND WARNING EXPLANATION MESSAGES ARE ARE WRITTEN TO
C     A DATA FILE FOR 2 REASONS:
C
C
C         1.  THE ERROR HANDLER SHOULD NOT IMMEDIATELY
C             WRITE INFORMATION OUT ON THE PS 300 SCREEN
C             SINCE THE EXPLANATORY TEXT DEFINING THE ERROR
C             OR WARNING CONDITION MAY BE TAKEN AS DATA BY
C             THE PS 300 AND THEREFORE WIND UP NOT BEING
C             DISPLAYED ON THE PS 300 SCREEN (AS IN THE
C             CASE OF A CATASTROPHIC DATA TRANSMISSION
C             ERROR).
C
C         2.  THE LOGGING OF ERRORS AND WARNINGS TO A
C             LOGFILE ALLOWS ANY ERRORS AND/OR WARNINGS
C             TO BE REVIEWED AT A LATER TIME.
C
C
C


       SUBROUTINE ERR ( ERRCOD )

C
C        PROCEDURAL INTERFACE ERROR HANDLER:
C

       INCLUDE    'PROCONST.FOR/NOLIST'
       INTEGER*4  ERRCOD
       INTEGER*4  PSVMSERR
       LOGICAL    FILOPN
       DATA       FILOPN /.FALSE./
       EXTERNAL   PSVMSERR, DETERH, PIDCOD

       IF (FILOPN) GOTO 1
C
C        OPEN ERROR FILE FOR LOGGING OF ERRORS:
C
         OPEN (UNIT=10, FILE='PROERROR.LOG', STATUS='NEW',
     &         DISP='KEEP', ORGANIZATION='SEQUENTIAL',
     &         ACCESS='SEQUENTIAL', CARRIAGECONTROL='LIST')
         FILOPN = .TRUE.
C      END IF
     1 CALL PIDCOD (ERRCOD)
       IF (ERRCOD .LT. 512) GOTO 3
         WRITE (10, *) 'PS-I-ATDCOMLNK:  ATTEMPTING TO '
     &                 // 'DETACH PS 300/HOST COMMUNICATIONS '
     &                 // 'LINK.'
C
C        WHEN WE ATTEMPT TO PERFORM THE DETACH, USE A
```

```
C       DIFFERENT ERROR HANDLER SO AS NOT TO GET CAUGHT
C       IN A RECURSIVE LOOP IF WE CONSISTENTLY GET AN
C       ERROR WHEN ATTEMPTING TO DETACH.
C
        CALL PDTACH (DETERH)
        CLOSE (UNIT=10)
        IF ((ERRCOD .LT. PSFPAF) .OR.
     &      (ERRCOD .GT. PSFPPF)) GOTO 2
C
C         IDENTIFY VMS ERROR IF THERE WAS ONE
C
          CALL LIB$STOP (%VAL (PSVMSERR ()))
          GOTO 3
C       ELSE
    2     STOP
C       END IF
C     END IF
    3 RETURN
      END



      SUBROUTINE DETERH (ERRCOD)

C
C       MAIN ERROR HANDLER DETACH ERROR HANDLER:
C

      INTEGER*4  ERRCOD
      EXTERNAL   PIDCOD

      WRITE (10, *) 'PS-I-ERRWARDET:  ERROR/WARNING '
     &           // 'TRYING TO DETACH '
     &           // 'THE COMMUNICATIONS'
      WRITE (10, *) 'LINK BETWEEN THE PS 300 AND THE HOST.'
      CALL PIDCOD (ERRCOD)
      RETURN
      END



      SUBROUTINE PIDCOD (ERRCOD)

C
C       PIDCOD: IDENTIFY PROCEDURAL INTERFACE COMPLETION
C               CODE.
C

      INCLUDE   'PROCONST.FOR/NOLIST'
      INTEGER*4  ERRCOD
      CHARACTER  VMSDEF*133, PIDEF*133
      INTEGER*4  PSVMSERR
      CHARACTER  MSSG1*55, MSSG2*67
      PARAMETER  (MSSG1 = 'PS-W-UNRCOMCOD:  PROCEDURAL '
     &                 // 'INTERFACE '
     &                 // '(GSR) COMPLETION ')
      EXTERNAL   PSVMSERR

      WRITE (10, *) 'PS-I-PROERRWAR:  PROCEDURAL '
     &           // 'INTERFACE WARNING/'
```

```
     &           // 'ERROR COMPLETION CODE WAS '
      WRITE (10, *) 'RECEIVED.'
      IF (ERRCOD .NE. PSWBNC) GOTO 1
        WRITE (10, *) 'PS-W-BADNAMCHR:  BAD CHARACTER '
     &              // 'IN NAME WAS '
     &              // 'TRANSLATED TO:  "_".'
        GOTO 1000
C     ELSE
    1 IF (ERRCOD .NE. PSWNTL) GOTO 2
        WRITE (10, *) 'PS-W-NAMTOOLON:  NAME TOO '
     &              // 'LONG.  NAME WAS '
     &              // 'TRUNCATED TO '
        WRITE (10, *) '256 CHARACTERS.'
        GOTO 1000
C     ELSE
    2 IF (ERRCOD .NE. PSWSTL) GOTO 7
        WRITE (10, *) 'PS-W-STRTOOLON:  STRING TOO '
     &              // 'LONG.  STRING '
     &              // 'WAS TRUNCATED '
        WRITE (10, *) 'TO 240 CHARACTERS.'
        GOTO 1000
C     ELSE
    7 IF (ERRCOD .NE. PSWAAD) GOTO 8
        WRITE (10, *) 'PS-W-ATTALRDON:  ATTACH '
     &              // 'ALREADY DONE.  '
     &              // 'MULTIPLE CALL TO PATTCH WITHOUT'
        WRITE (10, *) 'INTERVENING PDTACH CALL IGNORED.'
        GOTO 1000
C     ELSE
    8 IF (ERRCOD .NE. PSWAKS) GOTO 9
        WRITE (10, *) 'PS-W-ATNKEYSEE:  ATTENTION KEY '
     &              // 'SEEN (DEPRESSED).'
        CALL PIBMSP
        GOTO 1000
C     ELSE
    9 IF (ERRCOD .NE. PSWBGC) GOTO 10
        WRITE (10, *) 'PS-W-BADGENCHR:  BAD GENERIC '
     &              // 'CHANNEL CHARACTER.  BAD '
        WRITE (10, *) 'CHARACTER IN STRING SENT VIA:  '
     &              // 'PPUTGX  WAS TRANSLATED TO '
        WRITE (10, *) 'A BLANK.'
        CALL PIBMSP
        GOTO 1000
C     ELSE
   10 IF (ERRCOD .NE. PSWBSC) GOTO 11
        WRITE (10, *) 'PS-W-BADSTRCHR:  BAD '
     &              // 'CHARACTER IN STRING WAS '
     &              // 'TRANSLATED TO A BLANK.'
        CALL PIBMSP
        GOTO 1000
C     ELSE
   11 IF (ERRCOD .NE. PSWBPC) GOTO 12
        WRITE (10, *) 'PS-W-BADPARCHR:  BAD PARSER '
     &              // 'CHANNEL CHARACTER.  BAD '
     &              // 'CHARACTER IN STRING SENT TO'
        WRITE (10, *) 'PS 300 PARSER VIA:  PPUTP  '
     &              // 'WAS TRANSLATED TO A BLANK.'
        CALL PIBMSP
        GOTO 1000
C     ELSE
```

```
   12 IF (ERRCOD .NE. PSEIMC) GOTO 13
         WRITE (10, *) 'PS-E-INVMUXCHA:  INVALID '
     &                 // 'MULTIPLEXING CHANNEL '
     &                 // 'SPECIFIED IN CALL TO:'
         WRITE (10, *) 'PMUXCI, PMUXP, OR PMUXG.'
         GOTO 1000
C      ELSE
   13 IF (ERRCOD .NE. PSEIVC) GOTO 14
         WRITE (10, *) 'PS-E-INVVECCLA:  INVALID '
     &                 // 'VECTOR LIST CLASS '
     &                 // 'SPECIFIED'
         WRITE (10, *) 'IN CALL TO:  PVCBEG.'
         GOTO 1000
C      ELSE
   14 IF (ERRCOD .NE. PSEIVD) GOTO 15
         WRITE (10, *) 'PS-E-INVVECDIM:  INVALID '
     &                 // 'VECTOR LIST DIMENSION '
     &                 // 'SPECIFIED IN CALL TO'
         WRITE (10, *) 'PVCBEG.'
         GOTO 1000
C      ELSE
   15 IF (ERRCOD .NE. PSEPOE) GOTO 16
         WRITE (10, *) 'PS-E-PREOPEEXP:  PREFIX '
     &                 // 'OPERATOR CALL WAS '
     &                 // 'EXPECTED.'
         GOTO 1000
C      ELSE
   16 IF (ERRCOD .NE. PSEFOE) GOTO 17
         WRITE (10, *) 'PS-E-FOLOPEEXP:  FOLLOW '
     &                 // 'OPERATOR CALL WAS '
     &                 // 'EXPECTED.'
         GOTO 1000
C      ELSE
   17 IF (ERRCOD .NE. PSELBE) GOTO 18
         WRITE (10, *) 'PS-E-LABBLKEXP:  CALL TO '
     &                 // 'PLAADD OR PLAEND WAS '
     &                 // 'EXPECTED.'
         GOTO 1000
C      ELSE
   18 IF (ERRCOD .NE. PSEVLE) GOTO 19
         WRITE (10, *) 'PS-E-VECLISEXP:  CALL TO '
     &                 // 'PVCLIS OR PVCEND '
     &                 // 'WAS EXPECTED.'
         GOTO 1000
C      ELSE
   19 IF (ERRCOD .NE. PSEAMV) GOTO 20
         WRITE (10, *) 'PS-E-ATTMULVEC:  ATTEMPTED '
     &                 // 'MULTIPLE CALL '
     &                 // 'SEQUENCE TO PVCLIS IS NOT'
         WRITE (10, *) 'PERMITTED FOR BLOCK '
     &                 // 'NORMALIZED VECTORS.'
         GOTO 1000
C      ELSE
   20 IF (ERRCOD .NE. PSEMLB) GOTO 21
         WRITE (10, *) 'PS-E-MISLABBEG:  MISSING '
     &                 // 'LABEL BLOCK BEGIN CALL.  '
     &                 // 'CALL TO PLAADD OR PLAEND'
         WRITE (10, *) 'WITHOUT CALL TO:  PLABEG.'
         GOTO 100C
C      ELSE
```

```
   21 IF (ERRCOD .NE. PSEMVB) GOTO 22
        WRITE (10, *) 'PS-E-MISVECBEG:  MISSING '
     &              // 'VECTOR LIST BEGIN '
     &              // 'CALL.  CALL TO PVCLIS'
        WRITE (10, *) 'OR PVCEND WITHOUT CALL '
     &              // 'TO:  PVCBEG.'
        GOTO 1000
C      ELSE
   22 IF (ERRCOD .NE. PSENUN) GOTO 23
        WRITE (10, *) 'PS-E-NULNAM:  NULL NAME '
     &              // 'PARAMETER IS NOT ALLOWED.'
        GOTO 1000
C      ELSE
   23 IF (ERRCOD .NE. PSEBCT) GOTO 24
        WRITE (10, *) 'PS-E-BADCOMTYP:  BAD '
     &              // 'COMPARISON TYPE OPERATOR '
     &              // 'SPECIFIED IN '
        WRITE (10, *) 'CALL TO:  PIFLEV.'
        GOTO 1000
C      ELSE
   24 IF (ERRCOD .NE. PSEIFN) GOTO 25
        WRITE (10, *) 'PS-E-INVFUNNAM:  INVALID '
     &              // 'FUNCTION NAME.  '
     &              // 'ATTEMPTED PS 300'
        WRITE (10, *) 'FUNCTION INSTANCE FAILED '
     &              // 'BECAUSE THE NAMED '
     &              // 'FUNCTION CANNOT POSSIBLY'
        WRITE (10, *) 'EXIST.  THE FUNCTION NAME '
     &              // 'IDENTIFYING THE '
     &              // 'FUNCTION TYPE TO INSTANCE'
        WRITE (10, *) 'WAS LONGER THAN 256 CHARACTERS.'
        GOTO 1000
C      ELSE
   25 IF (ERRCOD .NE. PSENNR) GOTO 26
        WRITE (10, *) 'PS-E-NULNAMREQ:  NULL NAME '
     &              // 'PARAMETER IS '
     &              // 'REQUIRED IN OPERATE MODE'
        WRITE (10, *) 'CALL FOLLOWING A PPREF OR '
     &              // 'PFOLL PROCEDURE CALL.'
        GOTO 1000
C      ELSE
   26 IF (ERRCOD .NE. PSETME) GOTO 27
        WRITE (10, *) 'PS-E-TOOMANEND:  TOO '
     &              // 'MANY END_STRUCTURE CALLS '
     &              // 'INVOKED.'
        GOTO 1000
C      ELSE
   27 IF (ERRCOD .NE. PSENOA) GOTO 28
        WRITE (10, *) 'PS-E-NOTATT:  THE PS 300 '
     &              // 'COMMUNICATIONS LINK '
     &              // 'HAS NOT '
        WRITE (10, *) 'YET BEEN ESTABLISHED.  '
     &              // 'PATTCH HAS NOT BEEN '
     &              // 'CALLED OR FAILED.'
        GOTO 1000
C      ELSE
   28 IF (ERRCOD .NE. PSEODR) GOTO 29
        WRITE (10, *) 'PS-E-OVEDURREA:  AN '
     &              // 'OVERRUN OCCURRED DURING '
     &              // 'A READ OPERATION.'
```

```
          WRITE (10, *) 'THE SPECIFIED INPUT BUFFER '
         &              // 'IN CALL TO:  PGET  '
         &              // 'OR:  PGETW'
          WRITE (10, *) 'WAS TOO SMALL AND '
         &              // 'TRUNCATION HAS OCCURRED.'
          GOTO 1000
C       ELSE
   29   IF (ERRCOD .NE. PREICP) GOTO 38
   38   IF (ERRCOD .NE. PSEPDT) GOTO 39
          WRITE (10, *) 'PS-E-PHYDEVTYP:  MISSING '
         &              // 'OR INVALID PHYSICAL '
         &              // 'DEVICE TYPE'
          WRITE (10, *) 'SPECIFIER IN CALL TO PATTCH.'
          CALL PVAXSP
          GOTO 1000
C       ELSE
   39   IF (ERRCOD .NE. PSELDN) GOTO 40
          WRITE (10, *) 'PS-E-LOGDEVNAM:  MISSING '
         &              // 'OR INVALID LOGICAL '
         &              // 'DEVICE NAME'
          WRITE (10, *) 'SPECIFIER IN CALL TO PATTCH.'
          CALL PVAXSP
          GOTO 1000
C       ELSE
   40   IF (ERRCOD .NE. PSEADE) GOTO 41
          WRITE (10, *) 'PS-E-ATTDELEXP:  ATTACH '
         &              // 'PARAMETER STRING '
         &              // 'DELIMITER'
          WRITE (10, *) '"/" WAS EXPECTED.'
          CALL PVAXSP
          GOTO 1000
C       ELSE
   41   IF (ERRCOD .NE. PSFPAF) GOTO 42
          WRITE (10, *) 'PS-F-PHYATTFAI:  '
         &              // 'PHYSICAL ATTACH OPERATION '
         &              // 'FAILED.'
          GOTO 1000
C       ELSE
   42   IF (ERRCOD .NE. PSFPDF) GOTO 43
          WRITE (10, *) 'PS-F-PHYDETFAI:  PHYSICAL '
         &              // 'DETACH OPERATION '
         &              // 'FAILED.'
          GOTO 1000
C       ELSE
   43   IF (ERRCOD .NE. PSFPGF) GOTO 44
          WRITE (10, *) 'PS-F-PHYGETFAI:  PHYSICAL '
         &              // 'GET OPERATION FAILED.'
          GOTO 1000
C       ELSE
   44   IF (ERRCOD .NE. PSFPPF) GOTO 45
          WRITE (10, *) 'PS-F-PHYPUTFAI:  PHYSICAL '
         &              // 'PUT OPERATION FAILED.'
          GOTO 1000
C       ELSE
   45   IF (ERRCOD .NE. PSFBTL) GOTO 46
          WRITE (10, *) 'PS-F-BUFTOCLAR:  BUFFER '
         &              // 'TOO LARGE ERROR IN '
         &              // 'CALL TO:  PSPUT.'
          WRITE (10, *) 'THIS ERROR SHOULD NEVER '
         &              // 'OCCUR AND INDICATES A '
```

```
     &              // 'PROCEDURAL INTERFACE (GSR)'
        WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
        CALL PVAXSP
        GOTO 1000
C     ELSE
   46 IF (ERRCOD .NE. PSFWNA) GOTO 47
        WRITE (10, *) 'PS-F-WRONUMARG:  WRONG '
     &              // 'NUMBER OF ARGUMENTS '
     &              // 'IN CALL TO PROCEDURAL'
        WRITE (10, *) 'INTERFACE (GSR) LOW-LEVEL '
     &              // 'I/O PROCEDURE '
     &              // '(SOURCE FILE:  PROIOLIB.MAR).'
        WRITE (10, *) 'THIS ERROR SHOULD NEVER '
     &              // 'OCCUR AND INDICATES A '
     &              // 'PROCEDURAL INTERFACE (GSR)'
        WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
        CALL PVAXSP
        GOTO 1000
C     ELSE
   47 IF (ERRCOD .NE. PSFPTL) GOTO 48
        WRITE (10, *) 'PS-F-PROTOOLAR:  PROMPT '
     &              // 'BUFFER TOO LARGE '
     &              // 'ERROR IN CALL TO:  PSPRCV.'
        WRITE (10, *) 'THIS ERROR SHOULD NEVER '
     &              // 'OCCUR AND INDICATES A '
     &              // 'PROCEDURAL INTERFACE (GSR)'
        WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
        CALL PVAXSP
        GOTO 1000
C     ELSE
C
C     UNKNOWN ERROR MESSAGE ERROR MESSAGE.
C
   48 IF (ERRCOD .GE. 512) GOTO 49
        MSSG2 = MSSG1 // 'WARNING'
        GOTO 51
C     ELSE
   49   IF (ERRCOD .GE. 1024) GOTO 50
          MSSG2 = MSSG1 // 'ERROR '
          GOTO 51
C       ELSE
   50     MSSG2 = MSSG1 // 'FATAL ERROR '
C       END IF
C     END IF
   51 WRITE (10, *) MSSG2
      WRITE (10, *) 'CODE IS UNRECOGNIZED.'
      WRITE (10, *) 'PROBABLE PROCEDURAL '
     &            // 'INTERFACE (GSR) INTERNAL '
     &            // 'VALIDITY CHECK ERROR.'
C     END IF
 1000 IF ((ERRCOD .LT. PSFPAF) .OR.
     &    (ERRCOD .GT. PSFPPF)) GOTO 2000
        CALL PSFVMSERR ( VMSDEF, PIDEF )
        WRITE (10, *) 'DEC VAX/VMS ERROR '
     &              // 'DEFINITION IS:'
        WRITE (10, *)  VMSDEF
        WRITE (10, *) 'PROCEDURAL INTERFACE '
     &              // '(GSR) INTERPRETATION OF '
     &              // 'DEC VAX/VMS COMPLETION CODE:'
        WRITE (10, *)  PIDEF
```

```
          WRITE (10, *) 'DEC VAX/VMS ERROR CODE '
     &                  // 'VALUE WAS: ', PSVMSERR ()
C      END IF
 2000  WRITE (10, *)
       RETURN
       END



       SUBROUTINE PIBMSP

C
C        PIBMSP:  WRITE THE "IBM VERSION SPECIFIC"
C                 MESSAGE TO THE ERROR HANDLER FILE.
C

       WRITE (10, *) 'THIS ERROR/WARNING IS '
     &               // 'APPLICABLE ONLY TO THE IBM '
     &               // 'VERSION OF THE'
       WRITE (10, *) 'PROCEDURAL INTERFACE (GSR).'
       RETURN
       END



       SUBROUTINE PVAXSP

C
C        PVAXSP:  WRITE THE "DEC VAX/VMS VERSION
C                 SPECIFIC" MESSAGE TO THE ERROR
C                 HANDLER FILE.
C

       WRITE (10, *) 'THIS ERROR/WARNING IS '
     &               // 'APPLICABLE ONLY TO THE DEC '
     &               // 'VAX/VMS VERSION OF'
       WRITE (10, *) 'THE PROCEDURAL INTERFACE (GSR).'
       RETURN
       END
```

```
      SUBROUTINE INITBUF(IBUF,CHAN,IOSB)
C
C       THIS SUBROUTINE FILLS IBUF WITH DATA THAT DOES NOT CHANGE
C       DURING PROGRAM.  THIS DATA IS USED TO ADDRESS PS300 MEMORY
C       LOCATIONS.
C
        INTEGER*4 SYS$QIO,SYS$SETEF,SYS$WAITFR
        INTEGER*4 CHAN,STATUS,SYS$ASSIGN,SYS$QIOW,NAMADX(20)
        INTEGER*2 IBUF(3  ),NAMES(5,20),NAMADR(2,20),IOSB(4),BUFNUM
        CHARACTER*4 UNIT
        EQUIVALENCE (NAMADR,NAMADX)
        DATA NAMES/'TA','RG','T.','LO','OK','XY','ZV','C.','LO','OK',
     1             'GL','OB','E.','LO','OK','ST','AR','S.','LO','OK',
     1             'GL','OB','E.','TR','OT','ST','AR','S.','TR','OT',
     1             'GL','OB','E.','ER','OT','SA','TE','L.','RO','TX',
     1             'XY','PT','H.','TR','N1','XY','PT','H.','TR','N2',
     1             'XY','PT','H.','TR','N3','XY','PT','H.','TR','N4',
     1             'XY','PT','H.','TR','N5',
     1             'IN','FO','.R','AN','GE','IN','FO','.R','RA','TE',
     1             'IN','FO','.R','FU','EL','IN','FO','.R','TI','ME',
     1             'XY','PT','H.','SS','TR','XY','PT','H.','RO','TZ',
     1             'XY','PT','H.','RO','TX'/

        DATA UNIT/'PIA0'/
C
C       WAIT
C
        DO 5 J=1,100000
5       CONTINUE
C
C       SET UP HOUSEKEEPING
C
C       GET A CHANNEL NUMBER
C
        STATUS=SYS$ASSIGN(UNIT,CHAN,,)
        IF(STATUS.NE.1) THEN
          TYPE *,'BAD ASSIGN!  <STATUS> = ',STATUS
          STOP
        ENDIF
C
C       DETACH FOR SAFETY:       34 --> DETACH FUNCTION CODE
C
10      STATUS=SYS$QIOW(,%VAL(CHAN),%VAL(34),IOSB,,,,,,,,)
C
C       ATTACH:                  33 --> ATTACH FUNCTION CODE
C
        STATUS=SYS$QIOW(,%VAL(CHAN),%VAL(33),IOSB,,,,,,,,)
        IF(STATUS.NE.1) THEN
          TYPE *,'BAD ATTACH!  <STATUS> = ',STATUS
          STOP
        ENDIF
C
C       GET THE ADDRESSES OF THE ENTITIES TO UPDATE
C       43 --> LOOKUP NAMED ENTITIES FUNCTION CODE
C
20      DO 25 I=1,20
          STATUS=SYS$QIOW(,%VAL(CHAN),%VAL(43),IOSB,,,NAMES(1,I),
     1           %VAL(10),%VAL(1),,,)
C
          IF(STATUS.EQ.1.AND.IOSB(1).EQ.1.AND.
```

```
     1          (IOSB(3).OR.IOSB(4)).NE.0) GOTO 21
                TYPE *,'BAD ENTITY FETCH! <STAT,IOSB> -- ',STATUS,IOSB
                STOP
C
C               GET THE ADDRESS FROM OUT OF THE IO STATUS BLOCK (IOSB)
C
21              DO 24 J=1,2
                  NAMADR(J,I)=IOSB(J+2)
24              CONTINUE
25            CONTINUE
C
C
C             OFFSET THE ADDRESSES TO GET PAST THE FIRST THREE FIELDS
C             OF THE NODE WHICH WE DO NOT WANT TO CHANGE.
C
              DO 30 I=1,20
                 NAMADX(I)=NAMADX(I)+8
30            CONTINUE
C
C             OFFSET TEXT BY AN ADDITIONAL 16
C
              DO 31 I=14,17
              NAMADX(I)=NAMADX(I)+16
 31           CONTINUE


C
C             BUFFER 1  SETUP
C             TRANSLATION NEEDS  7 ELEMENTS
C             ROTATION NEEDS 19 ELEMENTS
C             LOOKAT NEEDS 28 ELEMENTS
C
              IBUF(1)          = 20              ! TWENTY BLOCKS
              IBUF(2)          = NAMADR(1,1)     ! BLOCK ONE ADDRESS - TARGT.LOOK
              IBUF(3)          = NAMADR(2,1)     ! BLOCK ONE ADDRESS
              IBUF(4)          = 27              ! WORD COUNT FOR BLOCK 1
              IBUF(24)         = 1               ! TRAN FLAG
              IBUF(32)         = NAMADR(1,2)     ! BLOCK TWO ADDRESS - XYZVC.LOOK
              IBUF(33)         = NAMADR(2,2)     ! BLOCK TWO ADDRESS
              IBUF(34)         = 27              ! WORD COUNT FOR BLOCK 2
              IBUF(54)         = 1               ! TRAN FLAG
              IBUF(62)         = NAMADR(1,3)     ! BLOCK 3 ADDRESS - GLOBE.LOOK
              IBUF(63)         = NAMADR(2,3)     ! BLOCK 3 ADDRESS
              IBUF(64)         = 27              ! WORD COUNT FOR BLOCK 3
              IBUF(84)         = 1               ! TRAN FLAG
              IBUF(92)         = NAMADR(1,4)     ! BLOCK 4 ADDRESS - STARS.LOOK
              IBUF(93)         = NAMADR(2,4)     ! BLOCK 4 ADDRESS
              IBUF(94)         = 27              ! WORD COUNT FOR BLOCK 4
              IBUF(114)        = 1               ! TRAN FLAG

              IBUF(122)        = NAMADR(1,5)     ! BLOCK 5 ADDRESS - GLOBE.TROT
              IBUF(123)        = NAMADR(2,5)     ! BLOCK 5 ADDRESS
              IBUF(124)        = 19              ! WORD COUNT FOR BLOCK 5
              IBUF(144)        = NAMADR(1,6)     ! BLOCK 6 ADDRESS - STARS.TROT
              IBUF(145)        = NAMADR(2,6)     ! BLOCK 6 ADDRESS
              IBUF(146)        = 19              ! WORD COUNT FOR BLOCK 6


              IBUF(166)        = NAMADR(1,7)     ! BLOCK 7 ADDRESS - GLOBE.ENOT
```

```
      IBUF(167)       = NAMADR(2,7)    ! BLOCK 7 ADDRESS
      IBUF(168)       = 19             ! WORD COUNT FOR BLOCK 7
      IBUF(188)       = NAMADR(1,8)    ! BLOCK 8 ADDRESS - SATEL.PX
      IBUF(189)       = NAMADR(2,8)    ! BLOCK 8 ADDRESS
      IBUF(190)       = 19             ! WORD COUNT FOR BLOCK 8

      IBUF(210)       = NAMADR(1,9)    ! BLOCK 9 ADDRESS - XYPTH.TRN1
      IBUF(211)       = NAMADR(2,9)    ! BLOCK 9 ADDRESS
      IBUF(212)       = 7              ! WORD COUNT FOR BLOCK 9
      IBUF(220)       = NAMADR(1,10)   ! BLOCK 10 ADDRESS - XYPTH.TRN2
      IBUF(221)       = NAMADR(2,10)   ! BLOCK 10 ADRESS
      IBUF(222)       = 7              ! WORD COUNT FOR BLOCK 10
      IBUF(230)       = NAMADR(1,11)   ! BLOCK 11 ADDRESS - XYPTH.TRN3
      IBUF(231)       = NAMADR(2,11)   ! BLOCK 11 ADRESS
      IBUF(232)       = 7              ! WORD COUNT FOR BLOCK 11
      IBUF(240)       = NAMADR(1,12)   ! BLOCK 12 ADDRESS - XYPTH.TRN4
      IBUF(241)       = NAMADR(2,12)   ! BLOCK 12 ADRESS
      IBUF(242)       = 7              ! WORD COUNT FOR BLOCK 12
      IBUF(250)       = NAMADR(1,13)   ! BLOCK 13 ADDRESS - XYPTH.TRN5
      IBUF(251)       = NAMADR(2,13)   ! BLOCK 13 ADRESS
      IBUF(252)       = 7              ! WORD COUNT FOR BLOCK 13
      IBUF(260)       = NAMADR(1,14)   ! BLOCK 14 ADDRESS - INFO.RANGE
      IBUF(261)       = NAMADR(2,14)   ! BLOCK 14 ADRESS
      IBUF(262)       = 3              ! WORD COUNT FOR BLOCK 14
      IBUF(266)       = NAMADR(1,15)   ! BLOCK 15 ADDRESS - INFO.RRATE
      IBUF(267)       = NAMADR(2,15)   ! BLOCK 15 ADRESS
      IBUF(268)       = 3              ! WORD COUNT FOR BLOCK 15
      IBUF(272)       = NAMADR(1,16)   ! BLOCK 16 ADDRESS - INFO.RFUEL
      IBUF(273)       = NAMADR(2,16)   ! BLOCK 16 ADRESS
      IBUF(274)       = 3              ! WORD COUNT FOR BLOCK 16
      IBUF(278)       = NAMADR(1,17)   ! BLOCK 17 ADDRESS - INFO.RTIME
      IBUF(279)       = NAMADR(2,17)   ! BLOCK 17 ADRESS
      IBUF(280)       = 3              ! WORD COUNT FOR BLOCK 17
      IBUF(284)       = NAMADR(1,18)   ! BLOCK 18 ADDRESS - XYPTH.SSTR
      IBUF(285)       = NAMADR(2,18)   ! BLOCK 18 ADRESS
      IBUF(286)       = 7              ! WORD COUNT FOR BLOCK 18
      IBUF(294)       = NAMADR(1,19)   ! BLOCK 19 ADDRESS - XYPTH.ROTZ
      IBUF(295)       = NAMADR(2,19)   ! BLOCK 19 ADRESS
      IBUF(296)       = 19             ! WORD COUNT FOR BLOCK 19
      IBUF(316)       = NAMADR(1,20)   ! BLOCK 20 ADDRESS - XYPTH.ROTX
      IBUF(317)       = NAMADR(2,20)   ! BLOCK 20 ADRESS
      IBUF(318)       = 19             ! WORD COUNT FOR BLOCK 20
C
      END
```

```
      SUBROUTINE IOBUFF (IFUNC,S,L,C,RESET)

      INTEGER S(24),L(24),C(12)
      INTEGER IFUNC,RESET
      INTEGER*2 DATAOUT,DATAIN
      INTEGER*2 DIBUF(10),DOBUF(10)
      INTEGER*4 ISTAT,ICALL
      INTEGER*4 NCHAN,DOFLAG,DIFLAG

      REAL ARATE,RATE

      DATA DOFLAG,DIFLAG/3,5/
      DATA NFRAME/10/
      DATA MODEOUT,MODEIN/8,7/
      DATA IUNIT/1/
      DATA ICHAN/1/
      DATA NCHAN/1/
      DATA ISMODE/0/
      DATA INIT/1/
      DATA RATE/80000.0/

      GO TO (100,200) IFUNC

100   CALL LPAIO (INIT,IUNIT,,RATE,,,,,ARATE,ISTAT,ICALL,,)
      IF (.NOT. ISTAT) GO TO 950
      RETURN

200   CONTINUE
      DATAOUT='0000'X
      DO 210 I=1,6
210   DATAOUT=DATAOUT+L(I)*2**(I-1)
      DO 220 I=1,NFRAME
220   DOBUF(I)=DATAOUT .XOR. 'FFFF'X
      CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYSSWAITFR (%VAL(DOFLAG))
230   CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 230
      CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYSSWAITFR (%VAL(DIFLAG))
240   CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 240
      DATAIN=(DIBUF(1).XOR.'FFFF'X)
      S(1)=DATAIN .AND. 1
      S(2)=(DATAIN .AND. 2)/2
      S(3)=(DATAIN .AND. 4)/4
      S(4)=(DATAIN .AND. 8)/8
      S(5)=(DATAIN .AND. 16)/16
      S(6)=(DATAIN .AND. 32)/32
      RESET=((64.XOR.DATAIN) .AND. 64)/64
      DATAOUT='4000'X
      DO 250 I=7,15
250   DATAOUT=DATAOUT+L(I)*2**(I-7)
      DO 260 I=1,NFRAME
260   DOBUF(I)=DATAOUT .XOR. 'FFFF'X
      CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
```

```
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DOFLAG))
270   CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 270
      CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DIFLAG))
280   CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 280
      DATAIN=(DIBUF(1).XOR.'FFFF'X)
      S(7)=DATAIN .AND. 1
      S(8)=(DATAIN .AND. 2)/2
      S(9)=(DATAIN .AND. 4)/4
      S(10)=(DATAIN .AND. 8)/8
      S(11)=(DATAIN .AND. 16)/16
      S(12)=(DATAIN .AND. 32)/32
      S(13)=(DATAIN .AND. 64)/64
      S(14)=(DATAIN .AND. 128)/128
      S(15)=(DATAIN .AND. 256)/256
      DATAOUT='8000'X
      DO 290 I=16,24
290   DATAOUT=DATAOUT+L(I)*2**(I-16)
      DO 300 I=1,NFRAME
300   DOBUF(I)=DATAOUT .XOR. 'FFFF'X
      CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DOFLAG))
310   CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 310
      CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DIFLAG))
320   CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 320
      DATAIN=(DIBUF(1).XOR.'FFFF'X)
      S(16)=DATAIN .AND. 1
      S(17)=(DATAIN .AND. 2)/2
      S(18)=(DATAIN .AND. 4)/4
      S(19)=(DATAIN .AND. 8)/8
      S(20)=(DATAIN .AND. 16)/16
      S(21)=(DATAIN .AND. 32)/32
      S(22)=(DATAIN .AND. 64)/64
      S(23)=(DATAIN .AND. 128)/128
      S(24)=(DATAIN .AND. 256)/256
      DATAOUT='C000'X
      DO 340 I=1,NFRAME
340   DOBUF(I)=DATAOUT .XOR. 'FFFF'X
      CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DOFLAG))
350   CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 350
      CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DIFLAG))
```

```
360     CALL LPAIO (17,,,,,,,,,,ISTAT,ICALL,LSTAT,)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 360
        DATAIN=(DIBUF(1).XOR.'FFFF'X)
        C(1)=(1.XOR.DATAIN) .AND. 1
        C(2)=((2.XOR.DATAIN) .AND. 2)/2
        C(3)=(DATAIN .AND. 4)/4
        C(4)=(DATAIN .AND. 8)/8
        C(5)=(DATAIN .AND. 16)/16
        C(6)=(DATAIN .AND. 32)/32
        C(7)=(DATAIN .AND. 64)/64
        C(8)=(DATAIN .AND. 128)/128
        C(9)=(DATAIN .AND. 256)/256
        C(10)=(DATAIN .AND. 512)/512
        C(11)=(DATAIN .AND. 1024)/1024
        C(12)=(DATAIN .AND. 2048)/2048
        RETURN

950     CONTINUE
        WRITE (5,1950) ISTAT,ICALL
1950    FORMAT (' ERROR IN CALL: STATUS = ',I6,' FROM CALL #',I6)
        CALL EXIT
        END
```

```
      SUBROUTINE LINTEG(X,OMEGA,A,DELTAT)
C
C     THIS IS A FIRST ORDER INTEGRATION SCHEME FOR TRANSLATIONAL
C     ACCELERATIONS IN THE REF (C-W) FRAME USING C-W EQUATIONS
C
      REAL X(6),XDOT(6),A(3),OMEGA,DELTAT
      INTEGER J
C
C       A(1-3) - X,Y,Z ACCELERATION
C       X(1-3) - X,Y,Z POSITION
C       X(4-6) - X,Y,Z VELOCITY
C       DELTAT - TIME STEP
C       OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH
C
      XDOT(1)=X(4)
      XDOT(2)=X(5)
      XDOT(3)=X(6)
C
C       BELOW ARE THE LINEARIZED EQUATIONS OF MOTION FOR AN INTERCEPT
C       VEHICLE RELATIVE TO A TARGET VEHICLE IN A CIRCULAR ORBIT
C       WITH KEPLARIAN MOTION
C
      XDOT(4)=A(1)-2.0*OMEGA*X(5)
      XDOT(5)=A(2)+3.0*OMEGA*OMEGA*X(2)+2.0*OMEGA*X(4)
      XDOT(6)=A(3)-X(3)*OMEGA*OMEGA
C
      DO 100 J=1,6
        X(J)=X(J)+DELTAT*XDOT(J)
100   CONTINUE
C
        END
```

```
      SUBROUTINE LOOK(T,X,OMEGA,TIME,FUEL,IBUF,CHAN,IOSB)
C
C       DETERMINES WHERE CAMERA IS POINTING (AT)
C       FROM SHUTTLE POSITION (X1,X2,X3) AND TRANSFORMS
C       FROM RH TO LH CARTESIAN COORDINATES (FM)
C       FOR VIEWING ON EVANS & SUTHERLAND PS300.
C       DETERMINES UP VECTOR FOR PS300 AND COMPUTES
C       EARTH AND STAR ROTATIONS AND POSITION OF HORIZON.
C       ALSO COMPUTES HEADS UP DISPLAY DATA.
C       INFORMATION IS THEN SENT TO THE PS300 FOR DISPLAY.
C
      INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'

      REAL CAM(3),T(3,3),X(6),MAT(4,4)
      REAL OMEGA,TIME,UP(3),RRATE,FUEL,RANGE,DT
      REAL*4 TROT,EROT,AT(3),FM(3),V(3),UPPS(3),RSAT
      REAL*4 XYPTH(3),NEWX(3),BMAT(3,3),BVEC(3),AMAT(3,3)
C
      INTEGER*4 CHAN,SYS$QIO,SYS$QIOW
      INTEGER*2 IOSB(4),IBUF(337)
C
C       COMPUTE SHUTTLE POSITION, RANGE AND RANGE RATE
C       WRT C-W FRAME (IN LH SYSTEM).
C
      FM(1)=X(1)
      FM(2)=X(2)
      FM(3)=-X(3)
C
      RANGE=SQRT(X(1)**2+X(2)**2+X(3)**2)
      RRATE=(X(1)*X(4)+X(2)*X(5)+X(3)*X(6))/RANGE
      IF (ABS(RRATE) .LT. .1) RRATE=0.0
C
C       COMPUTE SHUTTLE POSITION AND SCALE DOWN FOR
C       HUD XYPTH DISPLAY.
C
      XYPTH(1)=X(1)/100.
      XYPTH(2)=X(2)/100.
      XYPTH(3)=-X(3)/100.
C
C       ASSIGN VALUES TO CAMERA LOOK VECTOR IN BODY FRAME
C       (I. E. - LOOK UP OUT OF PAYLOAD BAY)
C
      CAM(1)=0.
      CAM(2)=0.
      CAM(3)=-1000.
C
C       TRANSFORM TO REF FRAME
C
      CALL BTOR(T,CAM,AT)
C
C       ADD SHUTTLE POSITION TO CAMERA VECTOR (REF FRAME)
C       AND CONVERT TO LEFT HANDED GRAPHICS COORDINATE SYSTEM.
C
      AT(1)=AT(1)+FM(1)
      AT(2)=AT(2)+FM(2)
      AT(3)=-AT(3)+FM(3)
C
C       ASSIGN VALUES TO UP VECTOR IN BODY FRAME.
C       (I. E. - OUT NOSE OF SHUTTLE)
C
```

```
      UP(1)=1000.
      UP(2)=0.
      UP(3)=0.
C
C       TRANSFORM TO REF FRAME, TRANSLATE TO 'AT' IN LH SYSTEM
C
      CALL BTOR(T,UP,UPPS)
C
      UPPS(1)=AT(1)+UPPS(1)
      UPPS(2)=AT(2)+UPPS(2)
      UPPS(3)=AT(3)-UPPS(3)
C
C       CONVERT AT,FM AND UPPS FROM FEET TO METERS
C
      AT(1)=AT(1)*.3048
      AT(2)=AT(2)*.3048
      AT(3)=AT(3)*.3048
      FM(1)=FM(1)*.3048
      FM(2)=FM(2)*.3048
      FM(3)=FM(3)*.3048
      UPPS(1)=UPPS(1)*.3048
      UPPS(2)=UPPS(2)*.3048
      UPPS(3)=UPPS(3)*.3048
C
C       DETERMINE EARTH ROTATION (DEG)
C
      EROT=TIME*.004178075
C
C       DETERMINE TARGET ROTATION (DEG)
C
      TROT= OMEGA*57.29577951*TIME
C
C       DETERMINE SATELLITE SPIN (DEG)
C
      RSAT=-TIME*30.0
C
C       SEND INFORMATION TO PS300
C
C
      CALL P919CV(XYPTH,3,IBUF(287))
      CALL ROT(90.,1,AMAT)
      CALL P919CV(AMAT,9,IBUF(319))
      CALL TLHT(T,AMAT)
      CALL P919CV(AMAT,9,IBUF(297))
C
      CALL ROT(RSAT,1,AMAT)
      CALL P919CV(AMAT,9,IBUF(191))
      CALL ROT(-EROT,2,AMAT)
      CALL P919CV(AMAT,9,IBUF(169))
      CALL ROT(TROT,3,AMAT)
      CALL P919CV(AMAT,9,IBUF(125))
      CALL P919CV(AMAT,9,IBUF(147))
C
      CALL LOOKAT(AT,FM,UPPS,BMAT,BVEC)
      CALL P919CV(BMAT,9,IBUF(5))
      CALL P919CV(BVEC,3,IBUF(25))
      CALL P919CV(BMAT,9,IBUF(35))
      CALL P919CV(BVEC,3,IBUF(55))

C       CALL PSNREA(RANGE*.01,1,'XYZVECSCL',ERR)
```

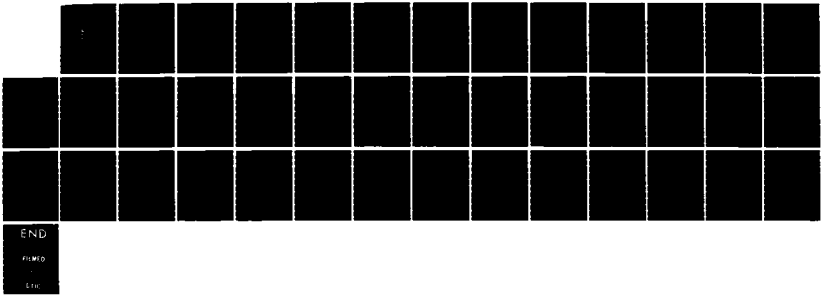
```
C
C       RESCALE TO DUS
C
        DO 57 I=1,3
            AT(I)=AT(I)/6378135.
            FM(I)=FM(I)/6378135.
            UPPS(I)=UPPS(I)/6378135.
57      CONTINUE
C
        CALL LOOKAT(AT,FM,UPPS,BMAT,BVEC)
        CALL P919CV(BMAT,9,IBUF(65))
        CALL P919CV(BVEC,3,IBUF(85))
        CALL P919CV(BMAT,9,IBUF(95))
        CALL P919CV(BVEC,3,IBUF(115))
C
        CALL NUMBER(RANGE,IBUF(263))
        CALL NUMBER(RRATE,IBUF(269))
        CALL NUMBER(FUEL,IBUF(275))
        CALL NUMBER(TIME,IBUF(281))
C
C     COMPUTE FUTURE POSITIONS FOR HEADS UP DISPLAY
C     AND SEND TO PS300 (DT IS TIME INCREMENT)
C
        DT=300.
        CALL PREDPATH(OMEGA,DT,X,NEWX)
        CALL P919CV(NEWX,3,IBUF(213))
        CALL PREDPATH(OMEGA,2.*DT,X,NEWX)
        CALL P919CV(NEWX,3,IBUF(223))
        CALL PREDPATH(OMEGA,3*DT,X,NEWX)
        CALL P919CV(NEWX,3,IBUF(233))
        CALL PREDPATH(OMEGA,4.*DT,X,NEWX)
        CALL P919CV(NEWX,3,IBUF(243))
        CALL PREDPATH(OMEGA,5.*DT,X,NEWX)
        CALL P919CV(NEWX,3,IBUF(253))

C       DO A WRITE SYNC
C       42          WRITE SYNC FUNCTION CODE
C       IOSB        IO STATUS BLOCK
C       IBUF        DATA BUFFER (ACTUALLY ADDRESS OF BUFFER, BY REFERENCE)
C       674         DATA BYTE COUNT (337 WORDS)
C       0           NOT CHARACTER DATA (1 = CHARACTER DATA)
C
C       SEND ALL DATA
C
        STATUS=SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(42),IOSB,,,
     +         IBUF(1),%VAL(674),%VAL(0),,,)
C
        IF (STATUS .NE. 1) THEN
          TYPE *,'BAD WRITE  <STATUS,IOSB>',STATUS,IOSB
          STOP
        ENDIF
C
        END
C
C
C
C
          SUBROUTINE ROT(ANGLE,IAXIS,AMAT)
C
C         ROUTINE TO GENERATE ROTATION REQUESTS TO PS300
```

```
C
C CALLING SEQUENCE:
C
C       CALL ROT(ANGLE,IAXIS,AMAT)
C
C WHERE:
C
C ANGLE IS THE REAL*4 ANGLE FOR ROTATION, IN DEGREES.  NEED NOT BE LIMITE
C       TO A SINGLE CIRCLE.
C IAXIS IS THE INTEGER*2 AXIS OF ROTATION (1=X, 2=Y, 3=Z).
C AMAT  IS THE REAL 3X3 MATRIX CALCULATED
C
C
      INTEGER*2 IAXIS,I,J
      REAL ANGLE,AMAT(3,3),PI180
      DATA PI180/0.017453/
C
      IDX(I)=MOD(I+2,3)+1     ! STATEMENT FUNCTION (NOT AN ARRAY)
C
      IF(IAXIS.LT.1.OR.IAXIS.GT.3) STOP 'PSLIB--AXIS OUT OF BOUNDS'
      DO 10 I=1,3
      DO 10 J=1,3
10    AMAT(I,J)=0.E0
      RADIAN = ANGLE * PI180
      C=COS(RADIAN)
      S=SIN(RADIAN)
      AMAT(IAXIS,IAXIS)=1.E0
      I=IDX(IAXIS-1)
      J=IDX(IAXIS+1)
      AMAT(I,I)=C
      AMAT(J,J)=C
      AMAT(I,J)=S
      AMAT(J,I)=-S
      RETURN
      END
C
C
C
C
      SUBROUTINE TLHT(T,AMAT)
C
C       ROUTINE TO CONVERT A RIGHT HANDED ROTATION MATRIX (T)
C       TO A LEFT HANDED ROTATION MATRIX (AMAT)
C
C CALLING SEQUENCE:
C
C       CALL TLHT(T,AMAT)
C
      REAL AMAT(3,3),T(3,3)
C
        AMAT(1,1)=T(1,1)
        AMAT(2,1)=T(2,1)
        AMAT(3,1)=-T(3,1)
        AMAT(1,2)=T(1,2)
        AMAT(2,2)=T(2,2)
        AMAT(3,2)=-T(3,2)
        AMAT(1,3)=-T(1,3)
        AMAT(2,3)=-T(2,3)
        AMAT(3,3)=T(3,3)
```

AD-A162 461 STS (SPACE TRANSPORTATION SYSTEM) TASK SIMULATOR(U) AIR FORCE ACADEMY CO S ALFANO 15 AUG 85 USAFA-TR-85-12 2/2

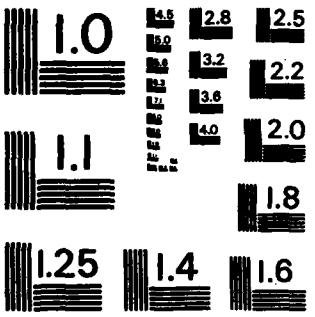
UNCLASSIFIED F/G 9/2 NL



END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

```
      RETURN
      END
C
C
C
C
      SUBROUTINE P919CV(MATRIX,N,BUFFER)
C
C       ROUTINE TO CONVERT A VAX REAL ARRAY TO ACP FLOATING-POINT FORMAT
C
C THIS ROUTINE CONVERTS AN ARRAY OF VAX SINGLE-PRECISION REAL NUMBERS INTO
  A NORMALIZED ARRAY OF 32-BIT ACP MANTISSAS, WITH THE ARRAY PRECEDED BY
C A 16-BIT EXPONENT.  THE MOST SIGNIFICANT ELEMENT IN THE ARRAY IS NORMAL-
C IZED.
C
C FORTRAN CALLING SEQUENCE:
C
C       CALL P919CV(MATRIX,N,BUFFER)
C
C WHERE:
C
C MATRIX IS AN N-ELEMENT REAL*4 ARRAY OF VAX FLOATING-POINT NUMBERS.
C N      IS THE INTEGER*2 SIZE OF ARRAY MATRIX.
C BUFFER IS THE INTEGER*2 ARRAY, OF LENGTH 2N+1, INTO WHICH THE RESULT IS
C       PLACED, WITH THE EXPONENT WORD FIRST, FOLLOWED BY THE ARRAY OF
C       32-BIT MANTISSAS.
C
C
      INTEGER*2 N,BUFFER(2*N+1),FWORD(2)
      REAL MATRIX(N),DMAX
      INTEGER*4 PSMEXP,PSMFRA,PSMNOR,DWORD,EDWORD
      EQUIVALENCE (DWORD,FWORD(1))
C     FIND LARGEST REAL NUMBER TO OBTAIN EXPONENT
      DMAX=0.
      DO 10 I=1,N
10    DMAX=AMAX1(DMAX,ABS(MATRIX(I)))
C     USE EXPONENT OF LARGEST NUMBER FOR NORMALIZATION
      DWORD=PSMEXP(DMAX)
      BUFFER(1)=FWORD(1)
      EDWORD=DWORD
C     OBTAIN NORMALIZED FRACTIONS AND LOAD INTO BUFFER
      DO 20 I=1,N
      DWORD=PSMNOR(MATRIX(I),EDWORD)
      BUFFER(2*I)=FWORD(2)
      BUFFER(2*I+1)=FWORD(1)
20    CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE LOOKAT(AT,FM,UP,MAT,VEC)
C
C       ROUTINE TO GENERATE LOOK AT, LOOK FROM,
C       LOOK UP REQUEST TO THE PS300
C
C       THIS ROUTINE UPDATES A PS300 DISPLAY LOOK NODE
C       WITH THE NECESSARY LOOK MATRIX
C
```

```
C       CALLING SEQUENCE:
C
C         CALL LOOKAT(AT,FM,UP,MAT,VEC)
C
C       WHERE:
C        INDEX IS THE INTEGER*2 NODE SUFFIX (1<=INDEX<=256)
C         WHICH CORRESPONDS TO ONE OF THE DISPLAY STRUCTURE
C         NODES N001 THROUGH N256.
C        AT, FM, UP : ARE THE VIEWING VECTORS
C        MAT : IS THE RESULTING 3X3 VIEWING MATRIX
C        BUFFER IS THE INTEGER*2 ARRAY, OF LENGTH 2N+1,
C         INTO WHICH THE RESULT IS PLACED, WITH THE EXPONENT
C         WORD FIRST, FOLLOWED BY THE ARRAY OF 32-BIT MANTISSAS.
C        VEC IS A 3 ELEMENT ARRAY CONTAINING ROW 4
C
C       COMPUTES 4X4 MATRIX FOR LOOK FUNCTION
C
      REAL AT(3),FM(3),UP(3),MAT(3,3),T(3),VEC(3)
      REAL D(3),E(3),M,F(3),G(3),H(3),MAG
C
      D(1)=AT(1)-FM(1)
      D(2)=AT(2)-FM(2)
      D(3)=AT(3)-FM(3)
C
      MAG=D(1)**2+D(2)**2+D(3)**2
      IF (MAG .GT. .1E-30) THEN
        MAG=SQRT(MAG)
       ELSE
        MAG=.1E-30
      ENDIF
C
      D(1)=D(1)/MAG
      D(2)=D(2)/MAG
      D(3)=D(3)/MAG
C
      E(1)=UP(1)-AT(1)
      E(2)=UP(2)-AT(2)
      E(3)=UP(3)-AT(3)
C
      M=D(1)*E(1)+D(2)*E(2)+D(3)*E(3)
C
      F(1)=E(1)-M*D(1)
      F(2)=E(2)-M*D(2)
      F(3)=E(3)-M*D(3)
C
      IF ((F(1)+F(2)+F(3)) .EQ. 0.0) THEN
        E(1)=0.0
        E(2)=1.0
        E(3)=0.0
        F(1)=E(1)-M*D(1)
        F(2)=E(2)-M*D(2)
        F(3)=E(3)-M*D(3)
        IF ((F(1)+F(2)+F(3)) .EQ. 0.0) THEN
          E(1)=0.0
          E(2)=0.0
          E(3)=1.0
          F(1)=E(1)-M*D(1)
          F(2)=E(2)-M*D(2)
          F(3)=E(3)-M*D(3)
        ENDIF
```

```
      ENDIF
C
      MAG=SQRT(F(1)**2+F(2)**2+F(3)**2)
C
      G(1)=F(1)/MAG
      G(2)=F(2)/MAG
      G(3)=F(3)/MAG
C
      H(1)=G(2)*D(3)-G(3)*D(2)
      H(2)=G(3)*D(1)-G(1)*D(3)
      H(3)=G(1)*D(2)-G(2)*D(1)
C
      T(1)=-FM(1)*H(1)-FM(2)*H(2)-FM(3)*H(3)
      T(2)=-FM(1)*G(1)-FM(2)*G(2)-FM(3)*G(3)
      T(3)=-FM(1)*D(1)-FP(2)*D(2)-FM(3)*D(3)
C
      MAT(1,1)=H(1)
      MAT(1,2)=H(2)
      MAT(1,3)=H(3)
      VEC(1)=T(1)
C
      MAT(2,1)=G(1)
      MAT(2,2)=G(2)
      MAT(2,3)=G(3)
      VEC(2)=T(2)
C
      MAT(3,1)=D(1)
      MAT(3,2)=D(2)
      MAT(3,3)=D(3)
      VEC(3)=T(3)
C
C
      RETURN
      END
C
C
C
C
      SUBROUTINE P920CV(MAT,VEC,BUFFER)
C
C     THIS SUBROUTINE FILLS IN A 4 X 4 ARRAY INTO THE BUFFER.
C     YOU LOAD A 3 X 4 MAT, AND AN ARRAY VEC(3) IS THE FOURTH ROW.
C     IT FILLS 34 ELEMENTS IN THE BUFFER.
C
      INTEGER*2 ITEMP1(9), ITEMP2(25), BUFFER(34)
      REAL MAT(3,4), VEC(4),MAT1(4,3)
      DO 100 I=1,3
      DO 99  J=1,4
        MAT1(J,I)=MAT(I,J)
 99   CONTINUE
 100  CONTINUE
      CALL P919CV(VEC,4,ITEMP1)
      CALL P919CV(MAT1,12,ITEMP2)
      BUFFER(1)  =  ITEMP1(1)
      DO 5 I = 1,25
         BUFFER(I+1) = ITEMP2(I)
  5   CONTINUE
      DO 10 J = 1,8
         BUFFER(J+26) = ITEMP1(J+1)
  10  CONTINUE
```

```
      RETURN
      END


       SUBROUTINE NUMBER(X,BUFFER)
C
C      THIS SUBROUTINE WILL TAKE ANY NUMBER FROM 9999.9 TO -999.9,
C      ENCODE IT, AND PUT IT IN THE PROPER PLACE IN THE OUTPUT
C      BUFFER, IBUF

       INTEGER*2 BUFFER(3), TBUF(3)
       CHARACTER*6 CHAR
       LOGICAL *1 TEMP0,TEMP1(6)
       EQUIVALENCE (CHAR,TEMP1,TBUF)

       IF(X.GT.9999.9) X=9999.9
       IF(X.LT.-999.9) X=-999.9
          ENCODE(6,201,CHAR)X
 201     FORMAT(F6.1)

          TEMP0=TEMP1(1)
          TEMP1(1)=TEMP1(2)
          TEMP1(2)=TEMP0
          TEMP0=TEMP1(3)
          TEMP1(3)=TEMP1(4)
          TEMP1(4)=TEMP0
          TEMP0=TEMP1(5)
          TEMP1(5)=TEMP1(6)
          TEMP1(6)=TEMP0
          BUFFER(1)=TBUF(1)
          BUFFER(2)=TBUF(2)
          BUFFER(3)=TBUF(3)
          RETURN
          END
C
```

```
C ***************************************************************************
C <<< LPAIO >>>> LPA11-K I/O ROUTINES; J.M. LIND; 03 MAY 83; REV C
C ***************************************************************************
C
      SUBROUTINE LPAIO (MODE,IUNIT,IFLAG,DRATE,ICHAN,NCHAN,NFRAME,IOBUF,
     1 ARATE,ISTAT,ICALL,LSTAT,ISMODE)
C
C REV A DESIGNED TO USE AST CALLS AT COMPLETION OF SWEEP (FILE NAME LPAIOA.
C REV B MODIFIED TO USE EVENT FLAGS INSTEAD OF AST CALLS 03 FEB 83 (JML)
C REV C MODIFIED TO ADD I/O MODE CALLING PARAMETER "ISMODE" AND TO ALLOW
C       FOR 2 CHANNEL DIGITAL I/O WITH THE ADDITION OF MODES 7,8,17, & 18.
C
C CALLING SEPCIFICATIONS:
C
C               MODE    MODE OF CALL WITH:
C
C                       MODE  = 1 INITIALIZE LPA11-K UNIT IUNIT
C                             = 3 ANALOG INPUT
C                             = 4 ANALOG OUTPUT
C                             = 5 DIGITAL INPUT A
C                             = 6 DIGITAL OUTPUT A
C                             = 7 DIGITAL INPUT B
C                             = 8 DIGITAL OUTPUT B
C                             =13 ANALOG INPUT STATUS
C                             =14 ANALOG OUTPUT STATUS
C                             =15 DIGITAL INPUT STATUS A
C                             =16 DIGITAL OUTPUT STATUS A
C                             =17 DIGITAL INPUT STATUS B
C                             =18 DIGITAL OUTPUT STATUS B
C
C               IUNIT   UNIT NUMBER OF THE DESIRED LPA11 SUBSYSTEM:
C
C                       IUNIT = 0 USES LAA0:
C                             = 1 USES LAB0:
C
C               IFLAG   NUMBER OF THE EVENT FLAG WHICH IS TO BE SET A COMPL
C               DRATE   DESIRED SAMPLE RATE (DO NOT EXCEED 80 KHZ)
C               ICHAN   START CHANNEL NUMBER
C               NCHAN   NUMBER OF CHANNELS (MUST BE 1 FOR DIGITAL I/O)
C               NFRAME  NUMBER OF FRAMES (NCHAN PER FRAME)
C               IOBUF   BUFFER FOR DATA (NFRAME * NCHAN 2 BYTE WORDS LONG)
C
C       RETURNED INFORMATION:
C
C               ARATE   ACTUAL SAMPLE RATE USED (0 FOR ERROR)
C               ISTAT   THREE WORD ARRAY WITH:
C
C                       ISTAT = 0  ERROR IN CALL
C                             = 1  SUCCCESFUL
C                             = X  VMS ERROR CODE
```

```
C                LSTAT   INTEGER*1 (BYTE) VARIABLE USED WITH ISTAT TO DEFI
C
C                        ISTAT   LSTAT   MEANING
C                        -------  -------  -------
C                        0        0       NORMAL - BUFFER 0 DONE
C                        -1       1       SWEEP TERMINATED OK
C                        -1       X       X = LPA11 ERROR CODE (USER'S GD P
C
C                ICALL   CALL NUMBER OF THIS PROGRAM (RELATES TO
C                        LPA11 I/O FUNCTION WHICH WAS LAST USED BEFORE
C                        RETURN TO THE CALLING PROGRAM)
C                        IF ICALL = 0, THEN MODE IS UNDEFINED!
C
C                ISMODE  SPECIFY MODE OF LPA11 SWEEP
C
C
C NOTES:         FOR MODES 5 - 8 (DIGITAL I/O), NCHAN MUST BE 1.
C                CHANNEL NUMBERING ALWAYS STARTS WITH 0.
C                IN MODE 1, THE SAMPLE RATE OF THE LPA11 CLOCK
C                IS SET, AND THE SAME RATE IS USED ON ALL LPA11 FUNCTIONS.
C                DRATE MUST NOT EXCEED 80 KHZ.  HOWEVER, THE LPA11 USER'S
C                MANUAL SPECIFIES MAXIMUM AGGREGRATE THROUGHPUT FOR MULTIR
C                ACTIVITIES AT 15 KHZ FOR ALL OPERATIONS COMBINED. (PARA 2
C
C WARNINGS:      WE SPECIFYING ISMODE = 512 IN THE DIGITAL INPUT MODE,
C                ONLY ONE CHANNEL OF DIGITAL I/O CAN BE USED AT A TIME.
C                OTHERWISE, THE PROGRAM WILL HANG WAITING FOR THE INPUT
C                FLAG TO BE SET BY THE LPA11.  WHEN USING ONLY ONE
C                CHANNEL OF DIGITAL I/O, THE ISMODE = 512 WILL WORK
C                PROPERLY.  IF ISMODE = 0 IS SPECIFIED FOR BOTH DIGIT
C                INPUT CHANNELS, THEN TWO CHANNELS MAY BE USED AT TH
C                SAME TIME.
C
C                WHEN USING A/D OR D/A MODE, YOU MUST SPECIFY AN ISMODE
C                OF 64 IN ORDER TO USE THE MULTIREQUEST MICROCODE WHICH
C                IS LOADED BY THIS ROUTINE.
C
C
C       1.  IN THE CASE OF DIGITAL OUTPUT, THE MODE OF THE LPA11 IN
C RUNNING THE DR11-K IS TO START OUTPUT IMMEDIATELY (THE MODE SPEICIFED
C IN THE CALL SHOULD BE ISMODE = 0.)
C
C       2.  IN THE CASE OF DIGITAL INPUT, THE MODE OF THE LPA11 IN
C RUNNING THE DR11-K IS TO START INPUT ON EXTERNAL TRIGGER (THE MODE
C SPECIFIED IN THE CALL SHOULD BE ISMODE = 512).  THE "EXTERNAL"
C TRIGGER IS ACTUALLY THE DR11-K "EXTERNAL DATA READY" LINE FOR THE
C EXTERNAL DEVICE.  (SEE DR11-K TIMING DIAGRAM ON PAGE 4-7 OF THE
C DR11-K INTERFACE USER'S GUIDE AND MAINTENANCE MANUAL.)
C IN THIS MODE, INTERRUPT WILL OCCURR ONLY AFTER THE EXTERNAL DEVICE
C CYCLES THE "EXTERNAL DATA READY" LINE.
C
C       3.  NOTE THAT THE CONFIGURATION OF THE DR11-K JUMPERS IS VERY
C IMPORTANT TO PROPER OPERATION OF THE DR11-K.  IN PARTICULAR, ALL
C S1 AND S2 SWITCHES SHOULD BE OFF TO DISABLE INTERRUPT BY TRANSITION
C OF THE DATA BITS (SEE TABLE 5-3).  IN ADDITION, JUMPERS W5 - W20
C MUST BE IN POSITION "B" IN ORDER TO READ DATA DIRECT FROM THE DATA
C INPUT LINES (AS OPPOSED TO THE BUFFER REGISTER INPUT).  THIS IS T
C DUE TO THE FACT THAT IN "BUFFER REGISTER" MODE, THE INDIVIDUAL
C DATA BITS IN THE BUFFER ARE SET ONLY ON TRANSITION OF THE DATA LINE
C (SEE PARAGRAPH 4-6 OF DR11-K INTERFACE USER'S GUIDE).  SINCE
```

```
C ALL SWITCHES ON S1 AND S2 ARE OFF, THE STATE OF JUMPERS W1-W4 IS A
C DON'T CARE.  JUMPERS W21-W23 SHOULD BE SET FOR APPROPRIATE POLARITY
C OF THE INTERNAL DATA ACCEPT AND INTERNAL DATA READY LINES.
C
C       4.  REMBER THAT ON DIGITAL OUTPUT, THIS PROGRAM SPECIFIES AT
C LEAST A 150 MICROSECOND DELAY BEFORE OUTPUT OF THE FIRST DIGITAL WORD
C (SEE PAGE 2-14 OF THE LPA11 USER'S GUIDE).  THIS IS NECESSARY
C IN ORDER TO ALLOW TIME FOR THE LPA11 TO RETRIEVE DATA FROM
C MEMORY.
```

```
C
C PARAMETER USAGE
C
C MODE   IUNIT   IFLAG   DRATE   ICHAN   NCHAN   NFRAME   IOBUF   ARA
C ----   -----   -----   -----   -----   -----   ------   -----   ---
C  1     X               X                                       X
C  3     X       X               X       X       X        X
C  4     X       X               X       X       X        X
C  5     X       X               X       X       X        X
C  6     X       X               X       X       X        X
C  7     X       X               X       X       X        X
C  8     X       X               X       X       X        X
C 13
C 14
C 15
C 16
C 17
C 18
C
C ***** PROGRAM DECLARATIONS *****
C
C VARIABLE DEFINITION SECTION
C
C       IOBUF   DATA BUFFER AREA (INTEGER*2)
C       XYBUF   LPA11 CONTROL BLOCK (50 LONGWORDS)
C       XYSTAT  LPA11 COMPLETION STATUS (FORM LPA$IGTBUF CALL)
C       IFLAG   LPA11 FLAG TO BE SET AT COMPLETION OF SWEEP
C       XYIOSB  I/O STATUS BLOCK FOR LPA11 (4 WORDS)
C       DXIOSZ  I/O STATUS BLOCK FOR DIGITAL I/O TO BUFFER Z (CH A OR B)
C       XYMSKB  LPA11 SUBSYSTEM MASKS AND NUM BUFFER
C       ISTAT   LPA11 STATUS LONGWORD
C       LSTAT   LPA11 I/O COMPLETION STATUS BYTE
C       NBUF    NUMBER OF BUFFERS TO BE FILLED (LONGWORD)
C
C WHERE "XY" IS AD FOR ANALOG-TO-DIGITAL
C               DA FOR DIGITAL-TO-ANALOG
C               DI FOR DIGITAL INPUT
C               DO FOR DIGITAL OUTPUT
C
C VARIABLE TYPE SPECIFICATIONS
C
        REAL            LPA$XRATE
        INTEGER*4       SYS$CLREF
C
        INTEGER*2       IOBUF(1),ADIOSB(4),DAIOSB(4)
        INTEGER*2       DIIOSA(4),DOIOSA(4),DIIOSB(4),DOIOSB(4)
        INTEGER*4       ADMSKB(2),DAMSKB(2)
        INTEGER*4       DIMSKA(2),DOMSKA(2),DIMSKB(2),DOMSKB(2)
        INTEGER*4       ISTAT,BUFNUM,NBUF,IFLAG
        BYTE            IDSC,IEMC,LSTAT
        INTEGER*2       IDSW,IEMW
C
C SET AREA FOR CONTRCL
        INTEGER*4       ADBUF(50),DABUF(50)
        INTEGER*4       DIBUFA(50),DOBUFA(50),DIBUFB(50),DOBUFB(50)
        EQUIVALENCE     (ADIOSB(1),ADBUF(1)),(DAIOSB(1),DABUF(1))
        EQUIVALENCE     (DIIOSA(1),DIBUFA(1)),(DOIOSA(1),DOBUFA(1))
        EQUIVALENCE     (DIIOSB(1),DIBUFB(1)),(DOIOSB(1),DOBUFB(1))
C
C
```

```
C ***** START OF PROGRAM *****
C DETERMINE MODE
C
C       LSTAT = 0
        GO TO (100,50,300,400,500,600,700,800,50,50,50,50,
     1 1300,1400,1500,1600,1700,1800),MODE
C EXECUTION STARTS HERE IF MODE IS UNDEFINED
 50     ICALL = 0
        ISTAT = 0
        GO TO 1950
C
C
C ***** MODE = 1 *****
C LOAD LPA11 SPECIFIED BY IUNIT WITH MICROCODE FOR MULTIREQUEST MODE
C
 100    CONTINUE
        ICALL = 101
        CALL LPASLOADMC (1,IUNIT,ISTAT,IERROR)
        IF (.NOT. ISTAT) GO TO 1950
C
C USE XRATE ROUTINE TO CALCULATE RATE AND PRESET VALUES FOR CLOCK A
C
C RATES ARE SUPPLIED/RETURNED BUT LPASXRATE REQUIRES INTERVALS
        AINTRVL = 1./DRATE
        ICALL = 102
        ACTUAL = LPASXRATE (AINTRVL,IRATE,IPRSET,0)
        ARATE  = 1./ACTUAL
C
C SET CLOCK RATE TO SPECIFIEC SAMPLE RATE (DO NOT EXCEED ABOUT 80 KHZ)
C
        ICALL = 103
        CALL LPASCLOCKA (IRATE,IPRSET,ISTAT,IUNIT)
        GO TO 1950
```

```
C ***** MODE = 3 *****
C START ANALOG-TO-DIGITAL INPUT SWEEP
C
 300    CONTINUE
C
C CLEAR A/D EVENT FLAG
        ICALL = 300
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C INITIALIZE ADBUF ARRAY FOR SWEEP
C
        ICALL = 301
        CALL LPA$SETIBF (ADBUF,ISTAT,ADMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
        ICALL = 302
        CALL LPA$LAMSKS (ADMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 303
        CALL LPA$RLSBUF (ADBUF,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C START A/D SWEEP BY SPECIFYING ONLY ONE BUFFER
C
        NPOINT = NFRAME * NCHAN
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 304
        CALL LPA$ADSWP (ADBUF,NPOINT,NBUF,ISMODE,,%VAL(IFLAG),,ICHAN,
     1 NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF A/D SWEEP CALL
        GO TO 1950
C
```

```
C ***** MODE = 4 *****
C START DIGITAL-TO-ANALOG OUTPUT SWEEP
C
 400    CONTINUE
C
C CLEAR D/A EVENT FLAG
        ICALL = 400
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C INITIALIZE DABUF ARRAY FOR SWEEP
C
        ICALL = 401
        CALL LPA$SETIBF (DABUF,ISTAT,DAMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
        ICALL = 402
        CALL LPA$LAMSKS (DAMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 403
        CALL LPA$RLSBUF (DABUF,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C CALCULATE NUMBER OF DATA POINTS
        NPOINT = NFRAME * NCHAN
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C IN D/A MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE.  SINCE THE LPA$XRATE CALL RETURNS THE V
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET.  (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
        LDELAY = 1
        IF (IRATE.EQ.1) LDELAY = 150
        IF (IRATE.EQ.2) LDELAY = 15
        IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
        IDWELL = 1
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 404
        CALL LPA$DASWP (DABUF,NPOINT,NBUF,ISMODE,IDWELL,%VAL(IFLAG),
     1 LDELAY,ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF D/A SWEEP CALL
        GO TO 1950
```

```
C ***** MODE = 5 *****
C START DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
 500    CONTINUE
C
C CLEAR DIGITAL INPUT EVENT FLAG
        ICALL = 500
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 501
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DIBUFA ARRAY FOR SWEEP
C
        ICALL = 502
        CALL LPA$SETIBF (DIBUFA,ISTAT,DIMSKA,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 503
        CALL LPA$LAMSKS (DIMSKA,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 504
        CALL LPA$RLSBUF (DIBUFA,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C START DIGITAL INPUT SWEEP BY SPECIFYING ONLY ONE BUFFER
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 505
        CALL LPA$DISWP (DIBUFA,NPOINT,NBUF,ISMODE,,%VAL(IFLAG),,
     1 ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT S   P C
        GO TO 1950
```

```
C ***** MODE = 6 *****
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
 600    CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
        ICALL = 600
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 601
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DOBUFA ARRAY FOR SWEEP
C
        ICALL = 602
        CALL LPA$SETIBF (DOBUFA,ISTAT,DOMSKA,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 603
        CALL LPA$LAMSKS (DOMSKA,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 604
        CALL LPA$RLSBUF (DOBUFA,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C
C IN DO MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE.  SINCE THE LPA$XRATE CALL RETURNS THE
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET.  (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
        LDELAY = 1
        IF (IRATE.EQ.1) LDELAY = 150
        IF (IRATE.EQ.2) LDELAY = 15
        IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
        IDWELL = 1
C PROCEED WITH SWEEP START CALL
```

```
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 605
        CALL LPA$DOSWP (DOBUFA,NPOINT,NBUF,ISMODE,IDWELL,%VAL(IFLAG)
     1 LDELAY,ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL OUTPUT SWEEP
        GO TO 1950
```

```
C ***** MODE = 7 *****
C START DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
 700    CONTINUE
C
C CLEAR DIGITAL INPUT EVENT FLAG
        ICALL = 700
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 701
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DIBUFB ARRAY FOR SWEEP
C
        ICALL = 702
        CALL LPA$SETIBF (DIBUFB,ISTAT,DIMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 703
        CALL LPA$LAMSKS (DIMSKB,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 704
        CALL LPA$RLSBUF (DIBUFB,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C START DIGITAL INPUT SWEEP BY SPECIFYING ONLY ONE BUFFER
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 705
        CALL LPA$DISWP (DIBUFB,NPOINT,NBUF,ISMODE,,%VAL(IFLAG),,
     1 ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT SWEEP CALL
        GO TO 1950
```

```
C ***** MODE = 8 *****
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
 800    CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
        ICALL = 800
        ISTAT = SYSSCLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 801
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DOBUFB ARRAY FOR SWEEP
C
        ICALL = 802
        CALL LPA$SETIBF (DOBUFB,ISTAT,DOMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 803
        CALL LPA$LAMSKS (DOMSKB,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 804
        CALL LPA$RLSBUF (DOBUFB,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
C
C IN DO MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFO
C THE FIRST CONVERSION TAKES PLACE.  SINCE THE LPA$XRATE CALL RETURNS TH
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET.  (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
        LDELAY = 1
        IF (IRATE.EQ.1) LDELAY = 150
        IF (IRATE.EQ.2) LDELAY = 15
        IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
        IDWELL = 1
```

```
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 805
        CALL LPASDOSWP (DOBUFB,NPOINT,NBUF,ISMODE,IDWELL,XVAL(IFLAG),
     1 LDELAY,ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL OUTPUT SWEEP CAL
        GO TO 1950
```

```
C ***** MODE = 13 *****
C GET STATUS OF A/D SWEEP
C
 1300   CONTINUE
C
        ICALL = 1301
        ISTAT = LPA$IWTBUF(ADBUF)
        LSTAT = IAND(ADIOSB(3),'FF00'X)/256
        GO TO 1950
C
C
C ***** MODE = 14 *****
C GET STATUS OF D/A SWEEP
C
 1400   CONTINUE
C
        ICALL = 1401
        ISTAT = LPA$IWTBUF(DABUF)
        LSTAT = IAND(DAIOSB(3),'FF00'X)/256
        GO TO 1950
C
C
C ***** MODE = 15 *****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
 1500   CONTINUE
C
        ICALL = 1501
        ISTAT = LPA$IWTBUF(DIBUFA)
        LSTAT = IAND(DIIOSA(3),'FF00'X)/256
        GO TO 1950
C
C
C ***** MODE = 16 *****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
 1600   CONTINUE
C
        ICALL = 1601
        ISTAT = LPA$IWTBUF(DOBUFA)
        LSTAT = IAND(DOIOSA(3),'FF00'X)/256
        GO TO 1950
C
C
C ***** MODE = 17 *****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
 1700   CONTINUE
C
        ICALL = 1701
        ISTAT = LPA$IWTBUF(DIBUFB)
        LSTAT = IAND(DIIOSB(3),'FF00'X)/256
        GO TO 1950
C
C
C ***** MODE = 18 *****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
 1800   CONTINUE
C
```

```
            ICALL = 1801
            ISTAT = LPASIWTBUF(DOBUFB)
            LSTAT = IAND(DOIOSB(3),'FF00'X)/256
            GO TO 1950
C
C
C
C ***** ERROR SERVICE ROUTINE *****
C
 1950   CONTINUE
C TRANSFER STATUS INFORMATION ON CALLING PARAMETER
C
            RETURN
C
            END
```

```
      SUBROUTINE LVLH(C,CO,Q,REF,DBX,DBY,DBZ,WB,JET,DELTAT,MAXWX,
     *  MAXWY,MAXWZ,PICK,T)
C
C       ATTEMPTS TO HOLD BODY AXES IN ALIGNMENT WITH LVLH AXES
C
      INTEGER C(12),CO(12),SNAP,I,J,PICK
      REAL Q(4),REF(4),YAW,PITCH,ROLL,DBX,DBY,DBZ
      REAL WB(3),JET(4,12),DELTAT,MAXWX,MAXWY,MAXWZ
      REAL ANG(3),T(3,3),PI,SWT,WT
      REAL FQ(4),FWB(3),FT(3,3),QT(3,3),REFT(3,3)
      REAL CONST,QQ(4)
C
      PI=2.0*ACOS(0.0)
C
C       WAS RHC JUST MOVED TO NEUTRAL ?  IF SO, TAKE SNAPSHOT.
C       REF(4) - QUATERNION AT TIME OF SNAPSHOT.
C       SNAP - FLAG TO TAKE SNAPSHOT OF QUATERNION.
C
      SNAP=0
      DO 150 I=7,12
        IF ((CO(I)-C(I)) .GT. 0) SNAP=1
150   CONTINUE
C
      IF (SNAP .EQ. 1) THEN
        REF(1)=Q(1)
        REF(2)=Q(2)
        REF(3)=Q(3)
        REF(4)=Q(4)
      ENDIF
C
C       COMPUTE FUTURE QUATERNION (NEXT ITERATION)
C         FWB - FUTURE BODY RATE
C         FQ - FUTURE QUATERNION
C
      ANG(1)=0.0
      ANG(2)=0.0
      ANG(3)=0.0
C
      FQ(1)=Q(1)
      FQ(2)=Q(2)
      FQ(3)=Q(3)
      FQ(4)=Q(4)
C
      FWB(1)=WB(1)
      FWB(2)=WB(2)
      FWB(3)=WB(3)
C
      CALL ROTATE(ANG,DELTAT,FWB,FQ,FT)
C
C       COMPUTE TRANSFORMATION FROM REF FRAME TO C-W FRAME
C
      CALL TRNSFM(REFT,REF)
C
C       COMPUTE TRANSFORMATION FROM FQ FRAME TO REF FRAME
C
      DO 81 I=1,3
        DO 82 J=1,3
          QT(I,J)=REFT(1,I)*FT(1,J)+REFT(2,I)*FT(2,J)+
     *            REFT(3,I)*FT(3,J)
82      CONTINUE
```

```
81    CONTINUE
C
C       COMPUTE QUATERNIONS (FROM FQ TO REF)
C
      QQ(4)=1.0+QT(1,1)+QT(2,2)+QT(3,3)
      IF (QQ(4) .LT. .1E-30) QQ(4)=.1E-30
      QQ(4)=SQRT(QQ(4))/2.0
      QQ(1)=(QT(3,2)-QT(2,3))/(4.0*QQ(4))
      QQ(2)=(QT(1,3)-QT(3,1))/(4.0*QQ(4))
      QQ(3)=(QT(2,1)-QT(1,2))/(4.0*QQ(4))
C
C       NORMALIZE QUATERNIONS
C
      CONST=SQRT(QQ(1)*QQ(1)+QQ(2)*QQ(2)+
     *       QQ(3)*QQ(3)+QQ(4)*QQ(4))
      QQ(1)=QQ(1)/CONST
      QQ(2)=QQ(2)/CONST
      QQ(3)=QQ(3)/CONST
      QQ(4)=QQ(4)/CONST
C
C       COMPUTE ANGLE OF ROTATION BETWEEN FRAMES
C
      WT=2.0*ACOS(QQ(4))
      SWT=SIN(WT/2.0)
C
C       COMPUTE ANGULAR DIFFERENCE (OF FQ W.R.T. REF)
C
      IF (WT .LT. .0000001) THEN
        ROLL=0.0
        PITCH=0.0
        YAW=0.0
       ELSE
        ROLL=QQ(1)*WT/SWT
        PITCH=QQ(2)*WT/SWT
        YAW=QQ(3)*WT/SWT
      ENDIF
C
C         DETERMINE THRUST PROFILE BY CHECKING ROLL, PITCH
C         AND YAW ANGLES/RATES AGAINST ATTITUDE HOLD CRITERIA.
C
      CALL CHECK(ROLL,DBX,WB(1),C(7),C(8),JET(PICK,7),
     *  JET(PICK,8),MAXWX,DELTAT)
      CALL CHECK(PITCH,DBY,WB(2),C(9),C(10),JET(PICK,9),
     *  JET(PICK,10),MAXWY,DELTAT)
      CALL CHECK(YAW,DBZ,WB(3),C(11),C(12),JET(PICK,11),
     *  JET(PICK,12),MAXWZ,DELTAT)
C
      END
```

```
      SUBROUTINE MODE(SA,SB,SC,LA,LB,LC)
C
C        TAKES ONE COLUMN OF DAP PANEL MANUAL MODE AND
C        FIGURES OUT WHAT LIGHTS TO TURN ON/OFF (LA,LB,LC)
C        USING SWITCH COMMANDS (SA,SB,SC)
C
      INTEGER SA,SB,SC,LA,LB,LC
C
      IF (SA .EQ. 1) THEN
        LA=1
        LB=0
        LC=0
      ENDIF
      IF (SB .EQ. 1) THEN
        LA=0
        LB=1
        LC=0
      ENDIF
      IF (SC .EQ. 1) THEN
        LA=0
        LB=0
        LC=1
      ENDIF
C
      END
```

```
      SUBROUTINE PREDPATH(OMEGA,TIME,OLDX,X)
C
C     THIS SUBROUTINES PREDICTS POSITION (X) RELATIVE TO
C     TARGET GIVEN PRESENT POSITION AND SPEED (OLDX) AND
C     FUTURE TIME (TIME)
C
      REAL OMEGA,X(3),OLDX(6),TIME,CW,SW
C
C
      CW=COS(OMEGA*TIME)
      SW=SIN(OMEGA*TIME)
C
      X(1)=OLDX(1)-2.0*OLDX(5)/OMEGA
      X(1)=X(1)-3.0*(OLDX(4)+2.0*OMEGA*OLDX(2))*TIME
      X(1)=X(1)+2.0*(3.0*OLDX(2)+2.0*OLDX(4)/OMEGA)*SW
      X(1)=X(1)+2.0*OLDX(5)*CW/OMEGA
C
      X(2)=4.0*OLDX(2)+2.0*OLDX(4)/OMEGA
      X(2)=X(2)-CW*(3.0*OLDX(2)+2.0*OLDX(4)/OMEGA)
      X(2)=X(2)+OLDX(5)*SW/OMEGA
C
      X(3)=OLDX(6)*SW/OMEGA+OLDX(3)*CW
C
C    SCALE DOWN FOR HUD DISPLAY
C    CONVERT TO LH SYSTEM
C
      X(1)=X(1)/100.
      X(2)=X(2)/100.
      X(3)=-X(3)/100.
C
      END
```

```
      SUBROUTINE PS300(ALTD,INCLIN)
C
C     THIS SUBROUTINE IS THE GRAPHICS PROGRAM FOR THE
C     SHUTTLE SIMULATOR. IT GENERATES A TARGET, ROTATING
C     EARTH, HORIZON AND STAR FIELD FOR BACKGROUND.
C     SUBROUTINE LOOK SENDS NEW DATA TO THIS PROGRAM IN
C     THE PS300 TO UPDATE TARGET RANGE AND ATTITUDE AND
C     EARTH/STAR ROTATIONS.
C
      INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'
C
      LOGICAL*1 POSLIN(73)
      REAL ALT,INCLIN,AT(3),FM(3),UP(3),DSTAR
      REAL HORR,HORT,HORD,LONG,ANG,ALTD
      REAL*4 V(3)
      INTEGER I,NVEC
      DIMENSION VECS(4,73)
C
      DATA POSLIN/.FALSE.,72*.TRUE./
C
C      ATTACH GRAPHICS DEVICE AND INITIALIZE GRAPHICS
C
      CALL PATTCH('LOGDEVNAM=PIA0:/PHYDEVTYP=PARALLEL',ERR)
      CALL PINIT(ERR)
C
C       V(3) - VECTOR ARRAY FOR PS300
C       HORR - RADIUS OF EARTH HORIZON AS SEEN FROM TARGET (DU)
C       HORD - DISTANCE OF HORIZON FROM EARTH CENTER (DU)
C       HORT - DISTANCE OF HORIZON FROM TARGET (DU)
C       DSTAR - DISTANCE OF STAR CLIPPING PLANE FROM TARGET (DU)
C
C       COMPUTE EARTH HORIZON RADIUS (HORR),
C       DISTANCE FROM EARTH CENTER (HORD),
C       AND DISTANCE FROM TARGET TO HORIZON (HORT)
C
      ALT=ABS(ALTD/6378.135)
      HORD=1.0/(1.0+ALT)
      HORR=SQRT(1.0-HORD*HORD)
      HORT=HORR/HORD
C
C     UNITS VARY FROM METERS TO DUS DEPENDING ON WHAT IS
C     BEING DISPLAYED. THIS UNIT JUGGLING IS DONE TO
C     MINIMIZE SCALING ERRORS INHERENT IN THE PS300.
C
C     INITIALIZE VECTORS FOR PS300 VIEWING
C      V - DUMMY VECTOR
C      AT,FM,UP - VIEWING VECTORS
C
      V(1)=0.0
      V(2)=0.0
      V(3)=0.0
      AT(1)=0.0
      AT(2)=0.0
      AT(3)=0.0
      FM(1)=0.0
      FM(2)=0.0
      FM(3)=-20.0
      UP(1)=0.0
      UP(2)=10.0
```

```
      UP(3)=0.0
C
C       COMPUTE ATTITUDE DISPLAY DATA FOR HUD
C
      CALL PBEGS('XYZVC',ERR)
C       UNITS ARE IN METERS
        CALL PSEDCL('CLIP',.FALSE.,'',ERR)
        CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
        CALL PFOV('FOV',35.0,1.0,3000.,'',ERR)
        CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
        CALL PSCALE('SCALE',V,'',ERR)
        CALL PSECOL('',180.0,1.0,'XARROW',ERR)
        CALL PSECOL('',180.0,1.0,'YARROW',ERR)
        CALL PSECOL('',180.0,1.0,'ZARROW',ERR)
      CALL PENDS(ERR)
C
C       COMPUTE PITCH AND FUTURE TRACK DATA FOR HUD
C
      CALL PBEGS('XYPTH',ERR)
        CALL PSEDCL('CLIP',.FALSE.,'',ERR)
        CALL PVIEWP('',.3,1.0,.3,1.0,1.0,1.0,'',ERR)
        CALL PWINDO('WNDW',-11.,11.,-11.,11.,0.,100.,'',ERR)
        CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
        CALL PROTY('ROTY',180.,'',ERR)
        CALL PSECOL('',300.0,1.0,'',ERR)
        CALL PTRANS('TRN1',V,'DIAMOND',ERR)
        CALL PTRANS('TRN2',V,'DIAMOND',ERR)
        CALL PTRANS('TRN3',V,'DIAMOND',ERR)
        CALL PTRANS('TRN4',V,'DIAMOND',ERR)
        CALL PTRANS('TRN5',V,'DIAMOND',ERR)
        CALL PSECOL('',180.0,1.0,'XYAXIS',ERR)
        CALL PTRANS('SSTR',V,'',ERR)
        CALL PROTZ('ROTZ',0.,'',ERR)
        CALL PROTX('ROTX',90.,'STICKSTS',ERR)
      CALL PENDS(ERR)
C
C       COMPUTE PIPPER DISPLAY DATA
C
      CALL PBEGS('GUNSITE',ERR)
        CALL PSEDCL('CLIP',.FALSE.,'',ERR)
        CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
        CALL PWINDO('WNDW',-1.,1.,-1.,1.,0.,100.,'',ERR)
        CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
        CALL PSECOL('',120.0,1.0,'PIPPER',ERR)
      CALL PENDS(ERR)
C
C       COMPUTE DISPLAY DATA FOR RANGE, RANGE RATE,
C       FUEL AND TIME.
C
      CALL PBEGS('INFO',ERR)
        CALL PSEDCL('CLIP',.FALSE.,'',ERR)
        CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
        CALL PWINDO('WNDW',0.,80.,0.,80.,0.,100.,'',ERR)
        CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
        CALL PSECHW('','',ERR)
        CALL PCHSCA('',1.,1.,'',ERR)
        CALL PCHS('',0.,79.,1.,1.,0.,'RANGE (FT)',ERR)
        CALL PCHS('',0.,77.,1.,1.,0.,'RANGE RATE',ERR)
        CALL PCHS('',0.,75.,1.,1.,0.,' FUEL USED',ERR)
        CALL PCHS('',0.,73.,1.,1.,0.,'TIME (SEC)',ERR)
```

```
          CALL PCHS('RANGE',12.,79.,1.,1.,0.,'000000',ERR)
          CALL PCHS('RRATE',12.,77.,1.,1.,0.,'000000',ERR)
          CALL PCHS('RFUEL',12.,75.,1.,1.,0.,'000000',ERR)
          CALL PCHS('RTIME',12.,73.,1.,1.,0.,'000000',ERR)
        CALL PENDS(ERR)
C
C         COMPUTE DISPLAY DATA FOR SPINNING SATELLITE
C
        CALL PBEGS('TARGT',ERR)
C         UNITS ARE IN METERS
          CALL PSEDCL('CLIP',.FALSE.,'*',ERR)
          CALL PVIEWP('*',-1.0,1.0,-1.0,1.0,1.0,1.0,'*',ERR)
          CALL PFOV('FOV',35.0,1.0,3000.,'*',ERR)
          CALL PLOOKA('LOOK',AT,FM,UP,'*',ERR)
          CALL PINST('*','SATEL',ERR)
        CALL PENDS(ERR)
C
C         CREATE A TDRS SATELLITE
C
        CALL PBEGS('SATEL',ERR)
            V(1)=-1.8
            V(2)=0.0
            V(3)=0.0
          CALL PTRANS('*',V,'*',ERR)
          CALL PROTX('ROTX',0.0,'*',ERR)
            V(1)=.2
            V(2)=.2
            V(3)=.2
          CALL PSCALE('*',V,'*',ERR)
          CALL PSECOL('*',300.0,1.0,'*',ERR)
          DO 150 I=10,360,10
            LONG=I
            CALL PROTX('*',LONG,'RTDRS',ERR)
150       CONTINUE
        CALL PENDS(ERR)
C
C
C         CREATE A SPINNING/INCLINED GLOBE OF THE EARTH,
C         ADD A STAR FIELD AND SET THE ENTIRE PICTURE
C         COUNTER-ROTATING W.R.T. TARGET ORBIT.
C         ADD A STATIONARY HORIZON
C
        CALL PBEGS('GLOBE',ERR)
C         UNITS ARE IN DU
          CALL PSEDCL('CLIP',.TRUE.,'*',ERR)
          CALL PVIEWP('*',-1.0,1.0,-1.0,1.0,1.0,1.0,'*',ERR)
          CALL PFOV('*',35.0,.1*ALT,
     *           HORT+.01,'*',ERR)
          CALL PLOOKA('LOOK',AT,FM,UP,'*',ERR)
            V(1)=0.0
            V(2)=(-1.-ALT)
            V(3)=0.0
          CALL PTRANS('*',V,'*',ERR)
          CALL PINST('*','HORIZON',ERR)
          CALL PROTZ('TROT',0.0,'*',ERR)
          CALL PSECOL('*',240.,1.,'*',ERR)
          CALL PROTX('*',90.-INCLIN,'*',ERR)
          CALL PROTY('EROT',0.0,'*',ERR)
          CALL PINST('*','WORLD',ERR)
          CALL PSECOL('*',240.,0.0,'*',ERR)
```

```
            CALL PINST('*','SPHERE',ERR)
            CALL PINST('*','LATLINE',ERR)
          CALL PENDS(ERR)
C
C         CREATE A HORIZON FROM A CIRCLE
C
          CALL PBEGS('HORIZON',ERR)
C           UNITS ARE IN DU
              V(1)=0.0
              V(2)=HORD
              V(3)=0.0
            CALL PTRANS('*',V,'*',ERR)
            CALL PROTX('*',90.,'*',ERR)
              V(1)=HORR
              V(2)=V(1)
              V(3)=V(1)
            CALL PSCALE('*',V,'CIRCLE',ERR)
          CALL PENDS(ERR)
C
C         CREATE A STAR FIELD
C
          CALL PBEGS('STARS',ERR)
C           UNITS ARE IN DU
            CALL PSEDCL('CLIP',.TRUE.,'*',ERR)
            CALL PVIEWP('*',-1.0,1.0,-1.0,1.0,1.0,1.0,'*',ERR)
              DSTAR=HORT+.9*SQRT((2.*ALT+1.)**2-1.)
            CALL PFOV('*',35.0,.1*DSTAR,DSTAR,'*',ERR)
            CALL PLOOKA('LOOK',AT,FM,UP,'*',ERR)
              V(1)=0.0
              V(2)=-1.-ALT
              V(3)=0.0
            CALL PTRANS('*',V,'*',ERR)
            CALL PROTZ('TROT',0.0,'*',ERR)
            CALL PSECOL('*',180.,0.,'*',ERR)
            CALL PROTY('*',90.,'STARS.TWINKLE',ERR)
            CALL PROTX('*',90.,'STARS.TWINKLE',ERR)
              V(1)=1.+2.*ALT
              V(2)=V(1)
              V(3)=V(1)
            CALL PSCALE('TWINKLE',V,'STAR',ERR)
          CALL PENDS(ERR)
C
C         BUILD A SPHERE FROM A CIRCLE
C
          CALL PBEGS('SPHERE',ERR)
            DO 200 I=10,180,10
              LONG=I
              CALL PROTY('*',LONG,'CIRCLE',ERR)
200         CONTINUE
          CALL PENDS(ERR)
C
C
C         BUILD A HEMI-SPHERE FROM A SEMI-CIRCLE
C
          CALL PBEGS('HEMI',ERR)
            DO 222 I=15,180,15
              LONG=I
              CALL PROTY('*',LONG,'SEMI',ERR)
222         CONTINUE
          CALL PENDS(ERR)
```

```
C
C
C       COMPUTE LINES OF LATITUDE USING CIRCLES
C
      CALL PBEGS('LATLINE',ERR)
        CALL PROTX('M',90.0,'CIRCLE',ERR)
        CALL PROTX('M',90.0,'LAT',ERR)
        CALL PROTX('M',-90.0,'LAT',ERR)
      CALL PENDS(ERR)
C
C
      CALL PBEGS('LAT',ERR)
        DO 20 I=10,80,10
           ANG=I*.0174532925
           V(1)=0.0
           V(2)=0.0
           V(3)=SIN(ANG)-SIN(ANG-.174532925)
         CALL PTRANS('M',V,'M',ERR)
           V(1)=COS(ANG)
           V(2)=V(1)
           V(3)=0.0
         CALL PSCALE('M',V,'CIRCLE',ERR)
20      CONTINUE
      CALL PENDS(ERR)
C
C
C       VECTOR LIST FOR CIRCLE
C
      DO 10 I=1,73
        ANG=5.0*(I-1)*.0174532925
        VECS(1,I)=COS(ANG)
        VECS(2,I)=SIN(ANG)
        VECS(3,I)=0.0
        VECS(4,I)=1.0
10    CONTINUE
C
      NVEC=73
      CALL PVCBEG('CIRCLE',NVEC,.TRUE.,.FALSE.,3,PVITEM,ERR)
        CALL PVCLIS(NVEC,VECS,POSLIN,ERR)
      CALL PVCEND(ERR)
C
C
C       VECTOR LIST FOR SEMI-CIRCLE
C
      DO 12 I=1,13
        ANG=(-90.0+15.0*(I-1))*.0174532925
        VECS(1,I)=SIN(ANG)
        VECS(2,I)=COS(ANG)
        VECS(3,I)=0.0
        VECS(4,I)=1.0
12    CONTINUE
C
      NVEC=13
      CALL PVCBEG('SEMI',NVEC,.TRUE.,.FALSE.,3,PVITEM,ERR)
        CALL PVCLIS(NVEC,VECS,POSLIN,ERR)
      CALL PVCEND(ERR)
C
C       COMMANDS FOR HUD ROTATION
C
C         SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
```

```
C
      CALL PSNST('  HUD  ',1,'DLABEL1',ERR)
      CALL PFN('HUD','YROTATE',ERR)
C
C         CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC1','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',1,1,'ACC1',ERR)
      CALL PSNREA(180.,2,'ACC1',ERR)
      CALL PSNREA(1.,3,'ACC1',ERR)
      CALL PSNREA(-15.,4,'ACC1',ERR)
      CALL PSNREA(360.,5,'ACC1',ERR)
      CALL PSNREA(0.,6,'ACC1',ERR)
C
C         CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION
C         AND THEN TO PROGRAM.
C
      CALL PCONN('ACC1',1,1,'HUD',ERR)
      CALL PCONN('HUD',1,1,'XYPTH.ROTY',ERR)
C
C        CALL NEEDED VECTOR LISTS
C
      CALL VECTOR ('DIAMOND',7)
      CALL VECTOR ('RTDRS',5)
      CALL VECTOR ('STICKSTS',8)
      CALL VECTOR ('XYAXIS',6)
      CALL VECTOR ('XARROW',6)
      CALL VECTOR ('YARROW',6)
      CALL VECTOR ('ZARROW',6)
      CALL VECTOR ('PIPPER',6)
      CALL VECTOR ('WORLD',5)
      CALL VECTOR ('STAR',4)
C
C
C       DISPLAY ALL
C
      CALL PDISP('XYPTH',ERR)
      CALL PDISP('GUNSITE',ERR)
      CALL PDISP('TARGT',ERR)
      CALL PDISP('GLOBE',ERR)
      CALL PDISP('STARS',ERR)
      CALL PDISP('INFO',ERR)
      CALL PDISP('XYZVC',ERR)
C
C       DETACH GRAPHICS DEVICE
C
      CALL PDTACH(ERR)
C
      END
```

```
      SUBROUTINE PULSE(COUNT1,COUNT2,C1,C2,DEL,JET1,JET2)
C
C     DETERMINES HOW MANY PULSES ARE NEEDED FOR THRUST COMMAND.
C     (COUNT1,COUNT2).  RESETS CONTROL SWITCHES (C1,C2) TO
C     EXECUTE PROPER NUMBER OF PULSES REGARDLESS OF THC/RHC
C     POSITION.
C
      INTEGER COUNT1,COUNT2,C1,C2
      REAL JET1,JET2,DEL
C
C       COUNT DOWN ONE PULSE
C
      COUNT1=COUNT1-1
      COUNT2=COUNT2-1
C
C       IF JET1 OR JET2 IS ZERO, EXIT SUBROUTINE
C
      IF (ABS(JET1) .LE. .00000001) GOTO 150
      IF (ABS(JET2) .LE. .00000001) GOTO 150
C
C       SET COUNTERS IF NOT PULSING FROM PREVIOUS COMMAND.
C
C       DEL - DESIRED VELOCITY INCREMENT
C       JET1,2 - VELOCITY INCREMENT GAINED BY ONE PULSE
C
      IF (COUNT1 .LT. 1) COUNT1=ABS(INT(C1*DEL/JET1))
      IF (COUNT2 .LT. 1) COUNT2=ABS(INT(C2*DEL/JET2))
C
C       SET CONTROL SWITCHES BASED ON COUNTERS
C
      IF (COUNT1 .GT. 0) C1=1
      IF (COUNT2 .GT. 0) C2=1
C
150   END
```

```
      SUBROUTINE QDOT(Q,WX,WY,WZ,DELTAT)
C
C     FIGURES QUATERNION RATE (QD) AND LINEARLY INTEGRATES
C     FOR TIME STEP DELTAT.
C
      REAL Q(4),QD(4),WX,WY,WZ,DELTAT,CONST
      INTEGER J
C
      QD(1)=(Q(2)*WZ-Q(3)*WY+Q(4)*WX)*.5
      QD(2)=(-Q(1)*WZ+Q(3)*WX+Q(4)*WY)*.5
      QD(3)=(Q(1)*WY-Q(2)*WX+Q(4)*WZ)*.5
      QD(4)=(-Q(1)*WX-Q(2)*WY-Q(3)*WZ)*.5
C
      DO 100 J=1,4
        Q(J)=Q(J)+DELTAT*QD(J)
100   CONTINUE
C
C       NORMALIZE QUATERNIONS
C
      CONST=SQRT(Q(1)*Q(1)+Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4))
      Q(1)=Q(1)/CONST
      Q(2)=Q(2)/CONST
      Q(3)=Q(3)/CONST
      Q(4)=Q(4)/CONST
C
      END
```

```
      SUBROUTINE ROTATE(ANG,DELTAT,WB,Q,T)
C
C     LINEARLY INTEGRATES TO FIND ROTATION IN BODY FRAME, THEN
C     TRANSFORMS TO REFERENCE FRAME.
C
      REAL ANG(3),WB(3),DELTAT,Q(4),T(3,3)
C
C       COMPUTE BODY RATES (WB)
C       (ANG IS ANGULAR ACCELERATION)
C
      WB(1)=WB(1)+DELTAT*ANG(1)
      WB(2)=WB(2)+DELTAT*ANG(2)
      WB(3)=WB(3)+DELTAT*ANG(3)
C
C       FIND QUATERNION RATE AND INTEGRATE
C
      CALL QDOT(Q,WB(1),WB(2),WB(3),DELTAT)
C
C       COMPUTE ROTATION MATRIX
C
      CALL TRNSFM(T,Q)
C
      END
```

```
      SUBROUTINE RTOB(T,REF,BOD)
C
C     TRANSFORMS FROM REF TO BOD FRAME GIVEN TRANSFORMATION MATRIX T
C
      REAL T(3,3),REF(3),BOD(3)
C
      BOD(1)=REF(1)*T(1,1)+REF(2)*T(2,1)+REF(3)*T(3,1)
      BOD(2)=REF(1)*T(1,2)+REF(2)*T(2,2)+REF(3)*T(3,2)
      BOD(3)=REF(1)*T(1,3)+REF(2)*T(2,3)+REF(3)*T(3,3)
C
      END
```

```
      SUBROUTINE SWITCH(S,L)
C
C        CONTROLS LIGHT PANEL FOR DAP BUTTONS
C
      INTEGER S(24),L(24),FILLER
C
C        S - SWITCH POSITION (1=TRIPPED,0=UNTRIPPED)
C        L - LIGHT (1=ON,0=OFF)
C        FILLER - EMPTY VARIABLE TO BE USED AS FILLER WHEN CALLING MODE
C
C        CHECK SELECTION MODE
C
      IF (S(1) .EQ. 1) L(1)=1
      IF (S(1) .EQ. 1) L(2)=0
      IF (S(2) .EQ. 1) L(1)=0
      IF (S(2) .EQ. 1) L(2)=1
C
C        CHECK AUTOPILOT MODE
C
      IF (S(3) .EQ. 1) L(3)=1
      IF (S(3) .EQ. 1) L(4)=0
      IF (S(4) .EQ. 1) L(3)=0
      IF (S(4) .EQ. 1) L(4)=1

C
C        CHECK THRUSTER MODE
C
      IF (S(5) .EQ. 1) L(5)=1
      IF (S(5) .EQ. 1) L(6)=0
      IF (S(6) .EQ. 1) L(5)=0
      IF (S(6) .EQ. 1) L(6)=1
C
C        CHECK X,Y,Z TRANSLATION MODES
C
C          CHECK LOW Z MODE:
C
      IF (S(17) .EQ. 1) THEN
        L(17)=1
        L(18)=0
        L(21)=0
        L(24)=0
      ENDIF
C
      CALL MODE(S(16),S(19),S(22),L(16),L(19),L(22))
      CALL MODE(0,S(20),S(23),FILLER,L(20),L(23))
      CALL MODE(S(18),S(21),S(24),L(18),L(21),L(24))
C
      IF ((L(18)+L(21)+L(24)) .GT. 0) L(17)=0
C
C          CHANGE THRUSTER MODE IF HI/LO Z SELECTED
C
      IF ((L(17)+L(18)) .GT. 0) THEN
        L(5)=1
        L(6)=0
      ENDIF
C
C        CHECK ROTATION MODES
C
      CALL MODE(S(7),S(10),S(13),L(7),L(10),L(13))
      CALL MODE(S(8),S(11),S(14),L(8),L(11),L(14))
```

```
      CALL MODE(S(9),S(12),S(15),L(9),L(12),L(15))
C
      END
```

```
      SUBROUTINE THRUST(JETSEL,A,ANG,GAS,C,PICK,TABLE)
C
C     TAKES JETSEL MATRIX AND SUMS ROWS AS COMMANDED BY C.
C     A - ACCEL X,Y,Z (FPS2,BODY FRAME)
C     ANG - ANGULAR ACCEL X,Y,Z (RAD/S2,BODY FRAME)
C     GAS - FUEL TAKEN FROM THREE TANKS
C           (1-FORWARD, 2-RIGHT AFT, 3-LEFT AFT)
C
      REAL JETSEL(44,9),A(3),ANG(3),GAS(3)
      INTEGER I,J,C(12),PICK,SELECT(44),ROW,TABLE(4,12,9)

C        DETERMINE WHAT THRUSTERS HAVE BEEN COMMANDED TO FIRE
C
      DO 145 I=1,44
        SELECT(I)=0
145   CONTINUE
C
      DO 175 I=1,12
        IF (C(I) .GT. 0) THEN
          DO 185 J=1,9
            ROW=TABLE(PICK,I,J)
            IF (ROW .GT. 0) SELECT(ROW)=1
185       CONTINUE
        ENDIF
175   CONTINUE
C
C        SUM COMMANDED ROWS OF JETSEL
C
      DO 95 I=1,3
        A(I)=0.0
        ANG(I)=0.0
        GAS(I)=0.0
95    CONTINUE
C
      DO 200 I=1,44
        IF (SELECT(I) .GT. 0) THEN
          A(1)=A(1)+JETSEL(I,1)
          A(2)=A(2)+JETSEL(I,2)
          A(3)=A(3)+JETSEL(I,3)
          ANG(1)=ANG(1)+JETSEL(I,4)
          ANG(2)=ANG(2)+JETSEL(I,5)
          ANG(3)=ANG(3)+JETSEL(I,6)
          GAS(1)=GAS(1)+JETSEL(I,7)
          GAS(2)=GAS(2)+JETSEL(I,8)
          GAS(3)=GAS(3)+JETSEL(I,9)
        ENDIF
200   CONTINUE
C
      END
```

```
      SUBROUTINE TRNSFM(T,Q)
C
C     COMPUTES TRANSFORMATION MATRIX (T) FROM QUATERNIONS (Q)
C
      REAL T(3,3),Q(4)
C
      T(1,1)=Q(1)*Q(1)-Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
      T(2,1)=2.0*(Q(1)*Q(2)+Q(3)*Q(4))
      T(3,1)=2.0*(Q(1)*Q(3)-Q(2)*Q(4))
      T(1,2)=2.0*(Q(1)*Q(2)-Q(3)*Q(4))
      T(2,2)=-Q(1)*Q(1)+Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
      T(3,2)=2.0*(Q(2)*Q(3)+Q(1)*Q(4))
      T(1,3)=2.0*(Q(1)*Q(3)+Q(2)*Q(4))
      T(2,3)=2.0*(Q(2)*Q(3)-Q(1)*Q(4))
      T(3,3)=-Q(1)*Q(1)-Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4)
C
      END
```

```
      SUBROUTINE UNITIZE(V)
C
C     UNITIZES V VECTOR
C
      REAL V(3),MAGV
C
      IF (ABS(V(1)) .LT. .1E-10) V(1)=.1E-10
      IF (ABS(V(2)) .LT. .1E-10) V(2)=.1E-10
      IF (ABS(V(3)) .LT. .1E-10) V(3)=.1E-10
      MAGV=SQRT(V(1)*V(1)+V(2)*V(2)+V(3)*V(3))
      V(1)=V(1)/MAGV
      V(2)=V(2)/MAGV
      V(3)=V(3)/MAGV
C
      END
```

```
      SUBROUTINE VECTOR (NAME,LENGTH)
C
C       THIS SUBROUTINE READS A VECTOR LIST FROM VAX
C       FILE AND PUTS IT IN USABLE FORM FOR THE PS300
C
      INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'

      INTEGER*4 IPOS, LENGTH, CLASS
      REAL*4 POINTS (4,2000)
      LOGICAL*1 POSLIN (2000), PL, VL
      CHARACTER NAME*8, FILENAME*26

      FILENAME='[ALFANO.DATA]'//NAME(:LENGTH)//'.DAT'
      OPEN ( UNIT=1, NAME=FILENAME,TYPE='OLD',READONLY)
      READ ( 1, 910) VL,IPOS
910   FORMAT ( A1,I10)
      IF ((VL.EQ.'C') .OR. (VL.EQ.'C'))  THEN
        CLASS=0
      ELSE IF ((VL.EQ.'D') .OR. (VL .EQ.'D')) THEN
        CLASS=1
      ELSE IF ((VL.EQ.'I') .OR. (VL .EQ.'I')) THEN
        CLASS=2
      ELSE IF ((VL.EQ.'S') .OR. (VL .EQ.'S')) THEN
        CLASS=3
      ENDIF
      DO 3410 I=1,IPOS
      READ ( 1, 911) PL,(POINTS(K,I),K=1,3)
911   FORMAT ( A1, 3F12.8)
      POINTS (4,I)=1
      POSLIN(I)=.FALSE.
      IF ((PL.EQ.'L') .OR.( PL.EQ.'L')) POSLIN(I)=.TRUE.
3410  CONTINUE
      CALL PVCBEG (NAME(:LENGTH),IPOS,.FALSE.,.FALSE.,3,CLASS,ERR)
       CALL PVCLIS (IPOS,POINTS,POSLIN,ERR)
      CALL PVCEND (ERR)
      CLOSE (UNIT=1)
      END
```

```
      SUBROUTINE WINDOW(T,TW,X1,X2,X3,TARG)
C
C     THIS SUBROUTINE COMPUTES THE TRANSFORMATION MATRIX
C     FROM REF TO WINDOW (TW) USING THE MATRIX FROM BOD
C     TO REF (T).  WINDOW IS ASSUMED TO HAVE -X,-Z BODY
C     COMPONENTS.  POSITION OF TARGET FROM WINDOW (TARG)
C     IS COMPUTED KNOWING POSITION OF SHUTTLE FROM TARGET
C     (X1,X2,X3).
C       WINDOW AXIS DEFINED AS FOLLOWS:
C         TARG(1) - INTO SCREEN
C         TARG(2) - LEFT
C         TARG(3) - UP
C
      REAL T(3,3),TW(3,3),X1,X2,X3,TARG(3),C
C
      C=SQRT(0.5)
C
C        COMPUTE TRANSFORMATION MATRIX
C
      TW(1,1)=-C*(T(1,1)+T(1,3))
      TW(1,2)=-C*(T(2,1)+T(2,3))
      TW(1,3)=-C*(T(3,1)+T(3,3))
      TW(2,1)=T(1,2)
      TW(2,2)=T(2,2)
      TW(2,3)=T(3,2)
      TW(3,1)=C*(T(1,1)-T(1,3))
      TW(3,2)=C*(T(2,1)-T(2,3))
      TW(3,3)=C*(T(3,1)-T(3,3))
C
C        COMPUTE TARGET VECTOR
C
      TARG(1)=-(X1*TW(1,1)+X2*TW(1,2)+X3*TW(1,3))
      TARG(2)=-(X1*TW(2,1)+X2*TW(2,2)+X3*TW(2,3))
      TARG(3)=-(X1*TW(3,1)+X2*TW(3,2)+X3*TW(3,3))
C
      END
```

END

FILMED

2-86

DTIC